

Validadores para CURP y RFC

Santos Zatarain Vera <santoszv(at)inftel.com.mx>

17 de mayo de 2016

Índice

1. Validación de CURP	2
1.1. Ejemplo 1	2
1.2. Ejemplo 2	3
1.3. Ejemplo 3	4
2. Validación de RFC	4
2.1. Ejemplo 1	5
2.2. Ejemplo 2	6
2.3. Ejemplo 3	6
3. Plantillas de los mensajes de error	7
3.1. ValidationMessages.properties	7
3.2. ValidationMessages_en.properties	7
4. Integración con Maven	7
5. Notas	8
6. Contacto y sugerencias	8
7. Licencia	8

1. Validación de CURP

Para validar la *Clave Única de Registro de Población* (CURP) se debe utilizar una o varias de las siguientes anotaciones:

1. `@mx.com.inftel.validators.curp.CURP`
2. `@mx.com.inftel.validators.curp.CURP.PalabraInconveniente`
3. `@mx.com.inftel.validators.curp.CURP.DigitoVerificador`

La anotación (1) valida si la cadena de texto contiene un CURP bien formado, la anotación (2) valida que no exista alguna palabra inconveniente al principio de la cadena de texto y por último, la anotación (3) calcula y valida el dígito verificador. Todas las anotaciones están implementadas como *regular constraints* según lo especificado en *JSR 349 Bean Validation Specification (formerly JSR 303)*.

Todas anotaciones tienen los parámetros comunes para *constraints*:

message La plantilla del mensaje de error

groups Los grupos de validación

payload Carga adicional del *constraint*

Los parámetros comunes usan los valores por defecto recomendados según lo especificado en *JSR 349 Bean Validation Specification (formerly JSR 303)*.

Solo la anotación (1) tiene dos parámetros adicionales a los comunes:

isNullValueValid Indicación para aceptar a `null` (nulo) como un valor válido, por defecto es `true`

isEmptyValueValid Indicación para aceptar a `""` (cadena vacía) como un valor válido, por defecto es `false`

Cada anotación contiene el subelemento `List` para especificar de forma múltiple el *constraint*, de acuerdo las recomendaciones del *JSR 349 Bean Validation Specification (formerly JSR 303)*.

1.1. Ejemplo 1

Validación simple, solo se comprueba que esté bien formado el CURP.

```

1  package my.pkg;
2
3  import mx.com.inftel.validators.curp.CURP;
4
5  public class JavaBean{
6
7      @CURP
8      private String curp;
9
10     public JavaBean(){
11
12         public String getCurp(){return curp;}
13
14         public void setCurp(String curp){this.curp = curp;}
15     }

```

1.2. Ejemplo 2

Validación de CURP, la cadena vacía también es aceptada como válida. Esta forma es usada por aplicaciones web principalmente, los formularios regularmente son enviados con campos vacíos.

```

1  package my.pkg;
2
3  import mx.com.inftel.validators.curp.CURP;
4
5  public class JavaBean{
6
7      @CURP(isEmptyValueValid = true)
8      private String curp;
9
10     public JavaBean(){
11
12         public String getCurp(){return curp;}
13
14         public void setCurp(String curp){this.curp = curp;}
15     }

```

1.3. Ejemplo 3

Validación de CURP, con comprobación del dígito verificador.

```
1 package my.pkg;
2
3 import mx.com.inftel.validators.curp.CURP;
4 import mx.com.inftel.validators.curp.CURP.DigitoVerificador;
5
6 public class JavaBean{
7
8     @CURP
9     @DigitoVerificador
10    private String curp;
11
12    public JavaBean(){
13
14        public String getCurp(){return curp;}
15
16        public void setCurp(String curp){this.curp = curp;}
17    }
```

2. Validación de RFC

Para validar el *Registro Federal de Contribuyente* (RFC) se debe utilizar una o varias de las siguientes anotaciones:

1. @mx.com.inftel.validators.rfc.RFC
2. @mx.com.inftel.validators.rfc.RFC.PalabraInconveniente
3. @mx.com.inftel.validators.rfc.RFC.DigitoVerificador

La anotación (1) valida si la cadena de texto contiene un RFC bien formado, la anotación (2) valida que no exista alguna palabra inconveniente al principio de la cadena de texto y por último, la anotación (3) calcula y valida el dígito verificador. Todas las anotaciones están implementadas como *regular constraints* según lo especificado en *JSR 349 Bean Validation Specification (formerly JSR 303)*.

Todas anotaciones tienen los parámetros comunes para *constraints*:

message La plantilla del mensaje de error
groups Los grupos de validación
payload Carga adicional del *constraint*

Los parámetros comunes usan los valores por defecto recomendados según lo especificado en *JSR 349 Bean Validation Specification (formerly JSR 303)*.

Solo la anotación (1) tiene cuatro parámetros adicionales a los comunes:

isNullValueValid Indicación para aceptar a `null` (nulo) como un valor válido, por defecto es `true`
isEmptyValueValid Indicación para aceptar a `""` (cadena vacía) como un valor válido, por defecto es `false`
isXAXX010101000ValueValid Indicación para aceptar a "XAXX010101000" (RFC para público general) como un valor válido, por defecto es `false`
isXEXX010101000ValueValid Indicación para aceptar a "XEXX010101000" (RFC para extranjero) como un valor válido, por defecto es `false`

Cada anotación contiene el subelemento `List` para especificar de forma múltiple el *constraint*, de acuerdo las recomendaciones del *JSR 349 Bean Validation Specification (formerly JSR 303)*.

2.1. Ejemplo 1

Validación simple, solo se comprueba que esté bien formado el RFC.

```
1 package my.pkg;
2
3 import mx.com.inftel.validators.rfc.RFC;
4
5 public class JavaBean{
6
7     @RFC
8     private String rfc;
9
10    public JavaBean(){
11
12        public String getRfc(){return rfc;}
13    }
```

```

14     public void setRfc(String rfc){this.rfc = rfc;}
15 }

```

2.2. Ejemplo 2

Validación con grupos, para validar también el dígito verificador, es necesario usar el grupo ConDigito.

```

1  package my.pkg;
2
3  import mx.com.inftel.validators.rfc.RFC;
4  import mx.com.inftel.validators.rfc.RFC.DigitoVerificador;
5
6  import javax.validation.groups.Default;
7
8  public class JavaBean{
9
10     @RFC(groups = {Default.class, ConDigito.class})
11     @DigitoVerificador(groups = {ConDigito.class})
12     private String rfc;
13
14     public JavaBean(){
15
16     public String getRfc(){return rfc;}
17
18     public void setRfc(String rfc){this.rfc = rfc;}
19 }

```

2.3. Ejemplo 3

Validación de RFC, la cadena vacía también es aceptada como válida, también es válido "XAXX010101000" y "XEXX010101000".

```

1  package my.pkg;
2
3  import mx.com.inftel.validators.rfc.RFC;
4  import mx.com.inftel.validators.rfc.RFC.DigitoVerificador;
5

```

```

6 public class JavaBean{
7
8     @RFC(isEmptyValueValid = true,
9         isXAXX010101000ValueValid = true,
10        isXEXX010101000ValueValid = true)
11    private String rfc;
12
13    public JavaBean(){ }
14
15    public String getRfc(){return rfc;}
16
17    public void setRfc(String rfc){this.rfc = rfc;}
18 }

```

3. Plantillas de los mensajes de error

Las plantillas *por defecto* están escritas en español. Para comodidad de los traductores, también se incluyen plantillas traducidas al inglés.

3.1. ValidationMessages.properties

```

1 mx.com.inftel.validators.curp.CURP.message=Este CURP está mal formado
2 mx.com.inftel.validators.curp.CURP.PalabraInconveniente.message=Este CURP contiene una palabra inconveniente
3 mx.com.inftel.validators.curp.CURP.DigitoVerificador.message=El dígito verificador no es correcto
4 mx.com.inftel.validators.rfc.RFC.message=Este RFC no está bien formado
5 mx.com.inftel.validators.rfc.RFC.PalabraInconveniente.message=Este RFC contiene una palabra inconveniente
6 mx.com.inftel.validators.rfc.RFC.DigitoVerificador.message=El dígito verificador no es correcto

```

3.2. ValidationMessages_en.properties

```

1 mx.com.inftel.validators.curp.CURP.message=This CURP is not well-formed
2 mx.com.inftel.validators.curp.CURP.PalabraInconveniente.message=This CURP has a drawback word
3 mx.com.inftel.validators.curp.CURP.DigitoVerificador.message=The check digit is not correct
4 mx.com.inftel.validators.rfc.RFC.message=This RFC is not well-formed
5 mx.com.inftel.validators.rfc.RFC.PalabraInconveniente.message=This RFC has a drawback word
6 mx.com.inftel.validators.rfc.RFC.DigitoVerificador.message=The check digit is not correct

```

4. Integración con Maven

Este proyecto está sincronizado en el *Maven Central Repository*, así que, para integrar las anotaciones a un proyecto *Maven*, solo se debe agregar la

dependencia, como lo muestra el siguiente *snippet*:

```
<dependency>
  <artifactId>mx.com.inftel.oss</artifactId>
  <groupId>curp-rfc-validators</groupId>
  <version>2.0</version>
</dependency>
```

5. Notas

Existen casos de RFC registrados ante el SAT con el dígito verificador incorrecto, por lo que es recomendable deshabilitar el *constraint* de validación del dígito verificador si es necesario.

6. Contacto y sugerencias

Es posible contactar directamente escribiendo un correo electrónico a Santos Zatarain Vera <[santoszv\(at\)inftel.com.mx](mailto:santoszv(at)inftel.com.mx)>, se agradece usar como asunto *Validadores CURP/RFC*.

7. Licencia

Copyright 2016 Santos Zatarain Vera

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.