

Dokumentation

“constraints”-Addon

Entwicklung eines Advanced Addons zur Erweiterung von Spring
Roo-Projekten um Crossfield-Validations-Constraints

*von Alex Kortsch und Fabian Schlötel
im Juli 2014*

Gliederung

1 Addon Dokumentation

- 1.1 Verwendete Technologie SpEL und geschaffene Voraussetzungen
- 1.2 Interfacedesign
- 1.3 Katalog für Simple Constraints

2 Beispielhafte Anwendung

- 2.1 Simple Constraint Command
- 2.2 Custom Raw Expression
- 2.3 Ergebnis

3 Fazit

- 3.1 Zusammenfassung
- 3.2 Herausforderungen
 - 3.2.1 Auflösen von Dependencies in Verbindung mit Roo-Cache
 - 3.2.2 Kompatibilität von Java 7 unter Apple OS X.9
 - 3.2.4 Annotation mit “.List” Element
- 3.3 Ausblick

1 Addon Dokumentation

1.1 Verwendete Technologie SpEL und geschaffene Voraussetzungen

Für die Validierung haben wir uns der Spring Expression Language bedient, welche eine dynamische und mächtige Validierung ermöglicht. Genauer wurde das Paket SpELAssert genutzt, um entsprechende Annotationen mit der SpEL-Expression auszuwerten.

Dies ermöglichte die Validierung mehrerer Attribute(Felder).

Um jedoch mit unserem “constraints”-Addon die entsprechenden SpELAssert Annotationen zu erzeugen, musste eine kleine Änderung im SpELAssert Paket durchgeführt werden. Dabei wurde in Zusammenarbeit durch Herrn Schmidt die Annotation SpELAssert mit dem internen “List” Attribut in die zwei separate Annotationen SpELAssertList, welche mehrerer Annotationen beinhalten kann, und für die Listenelemente die bekannte SpELAssert Annotation aufgeteilt. Eine zusätzliche Problembehandlung zur Vermeidung dieser Änderung wird in Kapitel 3.2.4 näher erläutert.

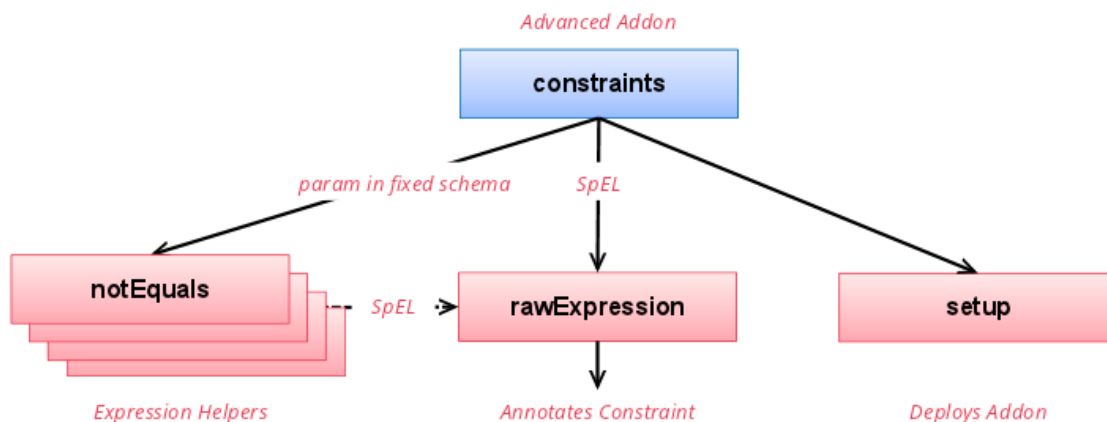
SpEL:

<http://docs.spring.io/spring-framework/docs/3.0.6.RELEASE/spring-framework-reference/html/expressions.html>

SpELAssert:

<https://github.com/jirutka/validator-spring>

1.2 Interfacedesign



1.3 Katalog für Simple Constraints

Type	Beispiel-Code	Bemerkung
setup	<pre>constraints setup</pre>	Dient zum registrieren des Addons.
equals	<pre>constraints equals --class <Object> --fieldlist <String> --message <String> --applyIf <SpEL Expression>*</pre>	Annotiert die Bedingung, dass unter den genannten Feldern (--fieldlist) keine gleichen Werte vorhanden sind.
notEquals	<pre>constraints notEquals --class <Object> --fieldlist <String> --message <String> --applyIf <SpEL Expression>*</pre>	Annotiert die Bedingung, dass unter den genannten Feldern (--fieldlist) ausschließlich gleiche Werte vorhanden sind.
contains	<pre>constraints contains --class <Object> --field <String> --fieldlist <String> --message <String> --applyIf <SpEL Expression>*</pre>	Annotiert die Bedingung, dass das genannte Feld (--field) die Werte der Felder aus --fieldlist stets beinhaltet.
notContains	<pre>constraints notContains --class <Object> --field <String> --fieldlist <String> --message <String> --applyIf <SpEL Expression>*</pre>	Annotiert die Bedingung, dass das genannte Feld (--field) die Werte der Felder aus --fieldlist nicht beinhaltet.
greater	<pre>constraints greater --class <Object> --field <String> --fieldlist <String> --message <String> --applyIf <SpEL Expression>*</pre>	Annotiert die Bedingung, dass der Wert von --field größer ist als die Werte der Felder aus --fieldlist.
smaller	<pre>constraints smaller --class <Object> --field <String> --fieldlist <String> --message <String> --applyIf <SpEL Expression>*</pre>	Annotiert die Bedingung, dass der Wert von --field kleiner ist als die Werte der Felder aus --fieldlist.
rawExpression	<pre>constraints rawExpression --class <Object> --expression <SpEL Expression> --message <String> --applyIf <SpEL Expression>* --helpers <Object>*</pre>	Annotiert eine beliebige valide SpEL-Expression.
removeAll	<pre>constraints removeAll --class <Object></pre>	Entfernt alle Annotationen aus einer Klasse.


removeRaw	constraints removeRaw --class <Object> --expression <SpEL Expression>	Entfernt eine einzelne Annotation anhand der passenden SpEL-Expression.
update	constraints update --class <Object> --expression <SpEL Expression> --message <String> --applyIf <SpEL Expression>* --helpers <Object>*	Aktualisiert die spezifizierte Annotation.

* nicht Pflicht

Anmerkung zur --fieldlist: Um ungewollte Eingabefehler zu vermeiden, wird die Eingabe der Feldnamen vor dem Annotieren überprüft. Dabei wird neben der Form der Eingabe hauptsächlich die Anzahl und die Existenz der Felder überprüft und ggf. entsprechende Fehlermeldungen zurückgegeben an den Benutzer.

2 Beispielhafte Anwendung

2.1 Simple Constraint Command



```

1.2.5.RELEASE [rev 8341dc2]

Welcome to Spring Roo. For assistance press TAB or type "hint" then hit ENTER.
roo> constraints setup
Updated ROOT/pom.xml [added dependencies com.constraints:com.constraints:0.1.0.
BUILD-SNAPSHOT, cz.jirutka.validator:validator-spring:1.0.2]
roo> constraints

constraints contains      constraints equals
constraints greater      constraints notContains
constraints notEquals    constraints rawExpression
constraints removeAll    constraints removeRaw
constraints setup        constraints smaller
constraints update

roo> constraints equals --

constraints equals --class      constraints equals --fieldlist
constraints equals --message

roo> constraints equals --class ~.model.Author --fieldlist "password,passwordCo
nfir" --message "Password confirmation failed!"
Field list is invalid "password,passwordConfir": Field "passwordConfir" doesn't
exists in class "Author"
roo> constraints equals --class ~.model.Author --fieldlist "password,passwordCo
nfir" --message "Password confirmation failed!"
Updated SRC_MAIN_JAVA/de/unileipzig/authors/model/Author.java
Create annotation with "password.equals(passwordConfirm)".
roo> 
```

2.2 Custom Raw Expression

```
constraints rawExpression --class ~.model.Author
    --expression "!title.contains(leadauthor.name)"
    --message "Title should be creative! Name of the author is trivial."
```

2.3 Ergebnis

```
@SpELAssertList({
    @cz.jirutka.validator.spring.SpELAssert(
        value = "!isBookAuthor.equals(isMagazineAuthor)",
        message = "You have to choose!")
    @cz.jirutka.validator.spring.SpELAssert(
        value = "password.equals(passwordConfirm)",
        message = "Password confirmation failed!")
})
```

3 Fazit

3.1 Zusammenfassung

Zusammenfassend für die erreichten Ziele, erkannten Herausforderungen und daraus entstandenen Einzelaufgaben, ist im Folgenden eine tabellarische Übersicht zu offenen und erledigten Aufgaben gegeben.

Offene Aufgaben	Erledigte Aufgaben
	<ul style="list-style-type: none">✓ Grundlegendes Addon erstellt✓ Shell Command-Konzept entwickeln✓ Beispielprojekt mit Authors und Books erzeugen✓ Validierung über mehr als 2 Attribute hinweg✓ SpELAssert einbinden und Annotation erzeugen (Projekt)✓ Bei mehreren Constraints, Neue in die SpELAssert Liste einfügen✓ Optionales --helpers in Commands umsetzen✓ Erfolgreiche Validierung zweier Attribute im Zielprojekt

	<ul style="list-style-type: none"> ✓ Diversifizierung der Validatoren ✓ Fehlerbehandlung bei falscher Handhabung ✓ Schutz vor doppelten Constraints ✓ Entfernen und aktualisieren eines einzelnen Constraints (via Raw) ✓ Test der Constraints auf unterschiedliche Datentypen
--	---

Im Vergleich zu der gestellten Aufgaben, ist davon auszugehen, dass mit der Erledigung der o.g. Einzelziele, die Aufgabe gelöst wurde.

3.2 Herausforderungen

Im Verlauf des Projekts sind einzelne Problemstellungen erkannt worden, die besondere Eigenschaften hatten. Im folgenden Teil werden diese Herausforderungen besonders hervorgehoben und erläutert.

3.2.1 Probleme beim Auflösen von Dependencies in Verbindung mit Roo-Cache

Im Zuge der Implementierung von Code-Artefakten zum Projekt, ist es zwischenzeitlich notwendig geworden manche Pakete als Import hinzuzuziehen. Hierbei ist aufgefallen, dass bei einem fälschlich eingebunden Paket, welches wieder entfernt wird, dieses noch im Roo-Cache bestehen bleibt als Import. Dadurch entsteht ein Fehler beim Registrieren des Addons bei OSGI. Durch das Löschen des Roo-Caches wurde dieser Fehler behoben, somit konnte das Addon wieder korrekt registriert werden.

3.2.2 Kompatibilität von Java 7 unter Apple OS X.9

Beim Installationsversuch von Spring Roo (getestet in den Versionen ab 1.2.2) ist aufgefallen, dass Roo nicht in Kombination mit Java Version 7 lauffähig installiert werden konnte. Da Java in der Version 7 jedoch zum Auslieferungsumfang des hier verwendeten Betriebssystems Apple OS X (rev. 9) gehört, war es nötig ein entsprechenden Down-Grade auf Java 6 vorzunehmen, um Roo lauffähig zu installieren.

3.2.4 Annotation mit “.List” Element

Bei der notwendigen Implementierung um multiple Annotationen durchzuführen, ist aufgefallen, dass - anders als erwartet - Code-Artefakte nicht ziele führend ge-rendert wurden.

Um eine Lösung zu erarbeiten, wurde unterstützend ein Stackoverflow-Eintrag zum Thema Angelegt. Eine Zielführende Antwort ist bisher weiterhin offen. Weitere Informationen zum Problem sind folgenden Foreneintrag zu entnehmen:

<http://stackoverflow.com/questions/24244764/spring-roo-add-multiple-cross-field-validation-annotation-list-jsr-303-bean-validation>

3.3 Ausblick

Mögliche Erweiterungen:

- Entfernen und Aktualisieren vereinfachen (Simple)
- Für bessere UX: --onField Option umsetzen (für Annotation)
- Weiterentwicklung der Vielfalt von Constraints (Simple)