Containerinhalt validieren

Diese Aktion liest einen Datensatz aus einem Container und validiert ihn anhand von Kriterien.

Parameter

idfield	der Request-Parameter, der die ID des Datensatzes enthält
list	die Liste, aus welcher der Datensatz entnommen wird (Container-ID oder Containername)
config	die Validierungsregeln alternativ auch Dateiname, wo die Regeln drin stehen

XML-Syntax der Validierungsregeln

```
<model>
<bind nodeset="Name" index="alias" required="true()" match="..." constraint="..."
type="xsd:anyURI"/>
<bind nodeset="Feld1" required="Feld2 = 'Wert'"/>
<bind nodeset="Location" .../>
</model>
```

Es kann auch ein Pfad angegeben werden, wo dann die Regeln hinterlegt sind (für mehr Flexibilität, da kann man die Regeln im Container speichern oder so)

Attribute

nodeset

gibt das Feld im Container an, welches validiert werden soll

index

ist ein Bezeichner für diesen Test (beliebiger Text)

required

`required="true()"` - Feld muß gefüllt sein

`required="Feld2 = 'Wert'"` - Feld muß in Abhängigkeit eines anderen Feldwertes gefüllt sein. Wichtig sind die Leerzeichen vor und nach dem `=`. Allerdings keine Leerzeichen vor und nach Feldname bzw. Wertangabe.

Beispiele

Feld muß gefüllt sein

```
...
<bind nodeset="Zaehlernummer" required="true()"/>
...
```

Die Zählernummer (z.B. eines Stromanschlusses) muß immer angegeben werden (im Container muß im entsprechenden Textfeld etwas stehen).

Feld muß in Abhängigkeit eines anderen Feldwertes gefüllt sein

```
...
<bind nodeset="Zaehlernummer" required="Neuanschluß = false"/>
...
```

Die Zählernummer muß angegeben werden, wenn im Formular das Häkchen bei "Neuanschluß" nicht gesetzt wurde (im Container angelegtes Häckchenfeld darf nicht angehakt sein).

Feld muß in Abhängigkeit eines anderen Feldwertes gefüllt sein

```
...
<bind nodeset="Zaehlernummer" required="Neuanschluß = 'nein'"/>
...
```

Gleiches Beispiel wie oben, nur daß das Feld "Neuanschluß" ein Textfeld ist und das Wort "nein" drin steht.

match

kann einen regulären Ausdruck enthalten (für einzeilige und mehrzeilige Textfelder)

Beispiel: E-Mail-Validierung

```
...
<bind nodeset="Email" index="2" match="^[_a-zA-Z0-9-](\.{0,1}[_a-zA-Z0-9-])*@([_a-zA-Z0-9-
]{2,}\.){0,}[_a-zA-Z0-9-]{3,}(\.[_a-zA-Z]{2,4}){1,2}$"/>
...
```

Der im Feld "Email" gespeicherte Wert muß dem angegebenen Regex ensprechen (dieser Regex ist nur einer von vielen).

constraint

kann eine Bedingung angegeben werden, die der Feldinhalt erfüllen muß (z.B. `. > 18:00`, zur Zeit `< >` für Datumsangaben, `< > =` für Zahlen und `=` für einzeiligen Text). Details siehe Abschnitt **constraint** unten.

Beispiel: Datum prüfen

```
...
<bind nodeset="Geburtsdatum" index="2" constraint="Geburtsdatum < #today"/>
...
```

Das Geburtsdatum muß vor heute liegen.

type

gibt den xsd-Typnamen an (zur Zeit noch nicht unterstützt)

constraint

Mit `constraint` werden Gültigkeitsbedingungen des Feldinhalts angegeben. Diese sind je nach Typ des Feldes unterschiedlich. Der Wert im Parameter `constraint` muß folgende Form haben:

```
<bind nodeset="..." constraint="{Containerfeld} {Operator} {Vergleichswert}" />
```

Wichtig sind die Leerzeichen, die die 3 Werte trennen.

Hier ist die Liste der unterstützten Bedingungen, geordnet nach Feldtyp:

Textfelder (einzeilig, mehrzeilig)

{Vergleichswert}: Gültig sind:

- Werte in Apostroph für Vergleich mit festen Wert
- Name eines anderen Feldes

- Spezialwerte
  - #null" (Feldinhalt ist null)
  - "#empty" (Feldinhalt ist Leerstring)
  - "#nodata" (Feldinhalt ist null oder Leerstring) (ab v2.5.8)

{Operator}:

- `=` prüft, ob Feld dem Testwert entspricht
- `!=` Prüft, ob Feld dem Testwert nicht entspricht (ab v2.5.8)
- `MATCH` Prüft, ob Feld dem regulären Ausdruck in Testwert entspricht (ab v2.5.8)

Datumsfelder

{Vergleichswert}

- Als Vergleichswert ist bisher (v2.5.8) nur ein fester Wert möglich. Er sollte wegen Kompatibilität zu zukünftigen Erweiterungen in Apostrophs eingeschlossen werden.
- Zeitfelder müssen in der Form "{Stunden}:{Minuten}[:{Sekunden}]" angegeben sein
- Datumsfelder müssen in der Form "{Tag}.{Monat}.{Jahr}" angegeben sein
- Datum-/Zeitfelder in der Form "{Tag}.{Monat}.{Jahr} {Stunden}:{Minuten}[:{Sekunden}]"
- Erweiterung mit ab v2.5.9: Schlüsselwort: "#today" (constraint="datum is #today+2")
- Erweiterung mit ab v2.5.9: Vergleich mit anderen Feld (constraint="vondatum before bisdatum" Bei zwei zu vergleichenden Feldern muß das Containerfeld als erstes angegeben werden.)

{Operator}

- `<` (`<`) - Prüft, ob Feld zeitlich vor dem Testwert liegt
- `>` (`>`) - Prüft, ob Feld zeitlich nach dem Testwert liegt
- "before" - Prüft, ob Feld zeitlich vor dem Testwert liegt (ab v2.5.9)
- "after" - Prüft, ob Feld zeitlich nach dem Testwert liegt (ab v2.5.9)
- "=" oder "is" - Prüft, ob Feld gleich dem Testwert ist (gleicher Tag) (ab v2.5.9)
- "lt" - Prüft, ob Feld zeitlich vor dem Testwert liegt (ab v2.5.9)
- "gt" - Prüft, ob Feld zeitlich nach dem Testwert liegt (ab v2.5.9)
- "beforeOrEqual" - Prüft, ob Feld zeitlich vor oder genau auf dem Testwert liegt (ab v2.5.9)
- "afterOrEqual" - Prüft, ob Feld zeitlich nach oder genau auf dem Testwert liegt (ab v2.5.9)
- "lte" - Prüft, ob Feld zeitlich vor oder genau auf dem Testwert liegt (ab v2.5.9)
- "gte" - Prüft, ob Feld zeitlich nach oder genau auf dem Testwert liegt (ab v2.5.9)

Zahlenfelder

{Vergleichswert}

- Als Vergleichswert ist bisher (v2.5.8) nur ein fester Wert möglich. Als Dezimaltrenner muß das Komma verwendet werden.

{Operator}

- `<` prüft, ob Feld kleiner als der Testwert ist
- `>` Prüft, ob Feld größer als der Testwert ist
- `=` Prüft, ob Feld gleich dem Testwert ist

Häkchenfelder (Wahrheitswert)

{Vergleichswert}

- 'j', 'ja', 'true', 'y', 'yes' für Test auf wahr
- '#null' für Test auf nicht gewählt (ab v2.5.8)
- alles andere für Test auf falsch

{Operator}

- `=` prüft, ob Feld dem Testwert entspricht
- `!=` prüft, ob Feld nicht dem Testwert entspricht (ab v2.5.8)

Verknüpfungen, Untercontainer

werden in Version 2.5.8 noch nicht unterstützt