

Actiereeksen: Tekstmanipulatie in Freemarker

Auteur: Rogier Visser (Laansloot IT)

Datum: 5 maart 2020

Naam: Laansloot-Actiereeksen-Freemarker-tekstmanipulatie.pdf

Versie: 0.1

Doelgroep: TOPdesk-applicatiebeheerders

Introductie

Actiereeksen binnen TOPdesk worden nog krachtiger wanneer je gebruik maakt van Freemarker. Met Freemarker kun je een extra laag logica inbouwen, waardoor je eindeloze mogelijkheden krijgt bij het maken van actiereeksen. Enige programmeerervaring is overigens wel gewenst.

Deze handleiding sluit aan bij het actiereeks-archief op het TOPdesk-forum van Laansloot IT. Het actiereeks-archief is een naslagwerk van actiereeksen waarmee je terugkerende taken binnen TOPdesk kunt automatiseren.

Je vindt het actiereeks-archief op de volgende locatie:

<https://topdeskforum.laansloot.nl/c/actiereeks-archief>

Disclaimer

Deze handleiding is puur van informatieve aard. Laansloot IT kan niet aansprakelijk gesteld worden voor enigerlei schade die zou kunnen voortkomen uit het volgen van deze handleiding. Laansloot IT geeft verder geen garantie dat actiereeksen doen wat ze moeten doen, en Laansloot IT geeft ook geen support hierop.

Heb je behoefte aan hulp? Stel dan je vraag op het TOPdesk-forum:

<https://topdeskforum.laansloot.nl/>

Copyright en licentie

Copyright op de teksten in deze handleiding ligt bij Laansloot IT. Copyright op de screenshots van TOPdesk ligt bij TOPdesk. Deze handleiding wordt gepubliceerd onder de licentie Creative Commons BY-NC-ND 4.0:

<https://creativecommons.org/licenses/by-nc-nd/4.0/deed.nl>

Waar gaat dit document precies over?

In het document "Laansloot-Actiereeksen-Freemarker-variabelen.pdf" is uitgelegd hoe je een variabele kunt aanmaken, en hoe je een bepaalde waarde toekent aan een variabele.

Dit document gaat één stap hierop verder. Je kunt namelijk met Freemarker ook extra bewerkingen laten toepassen op je variabele. Je kunt dus de waarde van een variabele aanpassen bij het aanroepen ervan.

Deze extra bewerkingen bewerkstellig je met de syntax `${variabele?extrabewerking}`. Dit document geeft een overzicht van de meest handige extra bewerkingen voor actiereeksen binnen TOPdesk.

Inhoud

- p.3: `?replace`
- p.6: `?keep_before`, `?keep_before_last`, `?keep_after`, `?keep_after_last`
- p.7: `?remove_beginning`, `?remove_ending`
- p.8: `?ensure_starts_with`, `?ensure_ends_with`
- p.9: Slicing
- p.11: Functies combineren
- p.12: Uit de praktijk: data uit een ingediend formulier opzoeken
- p.13: Regex (geavanceerd)

?replace

De meest basale extra bewerking is de ?replace-functie. De beginsituatie is één variabele die we simpelweg aanroepen:

JSON-body

```
<#assign omschr = "Kabelbreuk in Amsterdam">

{
  "action": "Korte omschrijving is ${omschr}"
}
```

Resultaat:



AUTOMATION,

Korte omschrijving is Kabelbreuk in Amsterdam

Met de ?replace-functie kunnen we waarde X laten vervangen door waarde Y:

JSON-body

```
<#assign omschr = "Kabelbreuk in Amsterdam">

{
  "action": "Korte omschrijving is ${omschr?replace("Amsterdam", "Rotterdam")}"
}
```

Resultaat:



AUTOMATION,

Korte omschrijving is Kabelbreuk in Rotterdam

De functies voor extra bewerkingen plaats je dus binnen de accolades {} van de variabele:

```
${omschr?replace("Amsterdam", "Rotterdam")}
```

Derde argument

De optie `?replace` vervangt standaard *alle* gevallen waar TOPdesk de oude waarde vindt in de string:

JSON-body

```
<#assign omschr = "Kabelbreuk in Amsterdam">

{
  "action": "Korte omschrijving is ${omschr?replace("a", "_")}"
}
```

Resultaat:



AUTOMATION,

Korte omschrijving is K_belbreuk in Amsterd_m

Wil je alleen in het eerste geval de oude waarde vervangen door de nieuwe waarde? Dan geef je als derde argument "f" op in de `?replace`-functie:

JSON-body

```
<#assign omschr = "Kabelbreuk in Amsterdam">

{
  "action": "Korte omschrijving is ${omschr?replace("a", "_", "f")}"
}
```

"f" betekent First only. Resultaat:



AUTOMATION,

Korte omschrijving is K_belbreuk in Amsterdam

Een ander derde argument binnen de `?replace`-functie is "i", waarmee je geen onderscheid maakt tussen hoofdletters en kleine letters. De syntax `${omschr?replace("a", "_", "i")}` wordt dan "K_belbreuk in _msterd_m".

In een overzichtje:

Syntax	Resultaat
<code>\${omschr?replace("a", "_")}</code>	K_belbreuk in Amsterd_m
<code>\${omschr?replace("a", "_", "f")}</code>	K_belbreuk in Amsterdam
<code>\${omschr?replace("a", "_", "i")}</code>	K_belbreuk in _msterd_m

Zoekwaarde verwijderen

Naast vervangen kun je met de `?replace`-functie ook tekst verwijderen, namelijk door de zoekwaarde te vervangen met een lege waarde. Je kunt op die manier bijvoorbeeld de streepjes in een UNID verwijderen:

JSON-body

```
<#assign omschr = unid>
```

```
{  
  "action": "https://locatietopdesk.net.com/tas/secure/incident?unid=${omschr?replace("-", "")}"  
}
```

Resultaat:



AUTOMATION,

[https://locatietopdesk.net.com/tas/secure/incident?
unid=db0057e430964cb1841a23f1abd7c922](https://locatietopdesk.net.com/tas/secure/incident?unid=db0057e430964cb1841a23f1abd7c922)

Overigens is `${unid}` een gereserveerde variabele in TOPdesk, en daarom kan `${omschr}` weggelaten worden:

JSON-body

```
{  
  "action": "https://locatietopdesk.net.com/tas/secure/incident?unid=${unid?replace("-", "")}"  
}
```

?keep_before, ?keep_after, ?keep_before_last, ?keep_after_last

De functies ?keep_before, ?keep_before_last, ?keep_after, ?keep_after_last gebruik je wanneer je een substring uit de variabele wil halen. In onderstaand voorbeeld scheiden we de substring op basis van een komma (dat we als karakter opgeven in de functie):

JSON-body

```
<#assign info = "Naam: Jan, Achternaam: Visboer, Leeftijd: 30, Beroep: Koopman ">
{
  "action": "Eerste deel is ${info?keep_before(",")}<br>Laatste deel is ${info?keep_after(",")}"
}
```

Resultaat:



AUTOMATION,

Eerste deel is Naam: Jan

Laatste deel is Achternaam: Visboer, Leeftijd: 30, Beroep: Koopman

De functies ?keep_before en ?keep_after scheiden zo dus de variabele \${info} bij de eerste komma die ze tegenkomen. De functies ?keep_before_last en ?keep_after_last scheiden juist bij de laatste komma:

JSON-body

```
<#assign info = "Naam: Jan, Achternaam: Visboer, Leeftijd: 30, Beroep: Koopman ">
{
  "action": "Eerste deel is ${info?keep_before_last(",")}<br>Laatste deel is ${info?keep_after_last(",")}"
}
```

Resultaat:



AUTOMATION,

Eerste deel is Naam: Jan, Achternaam: Visboer, Leeftijd: 30

Laatste deel is Beroep: Koopman

De komma is hier slechts een voorbeeld. Je kunt scheiden op elke waarde die je kiest. Wat zou er gebeuren wanneer we scheiden op "Visboer"?

```
${info?keep_before("Visboer")}
${info?keep_after("Visboer")}
```

?remove_beginning, ?remove_ending

Wil je specifiek waarden aan het begin of aan het eind van een variabele verwijderen? Dan gebruik je daarvoor de functies ?remove_beginning of ?remove_ending.

Binnen de functie geef je de zoekwaardes op die je wil weghalen, zoals "in: " of "!":

JSON-body

```
<#assign info = "in: 10, out: 20!">
```

```
{  
  "action": "${info?remove_beginning("in: ")}<br>${info?remove_ending("!")}"  
}
```

Resultaat:

A **AUTOMATION,**
10, out: 20!
in: 10, out: 20

Als de zoekwaardes niet voorkomen, dan doen de functies niets:

JSON-body

```
<#assign info = "in: 10, out: 20!">
```

```
{  
  "action": "${info?remove_beginning("X")}<br>${info?remove_ending("Y")}"  
}
```

Resultaat:

A **AUTOMATION,**
in: 10, out: 20!
in: 10, out: 20!

?ensure_starts_with, ?ensure_ends_with


Met deze functies dwing je af dat de ingevoegde waarde begint of eindigt met een bepaalde string. In onderstaand voorbeeld dwingen we de string "Naam: " af

JSON-body

```
<#assign info = "Pietje Bel">

{
  "action": "${info?ensure_starts_with("Naam: ")}"
}
```

Resultaat:

 **AUTOMATION,**
Naam: Pietje Bel


Als de opgegeven string al in de variabele zit, dan doet de functie in feite niets:

JSON-body

```
<#assign info = "Naam: Pietje Bel">

{
  "action": "${info?ensure_starts_with("Naam: ")}"
}
```

Het resultaat is dan precies hetzelfde:

 **AUTOMATION,**
Naam: Pietje Bel

De functie ?ensure_ends_with werkt op exact dezelfde manier, alleen dan met het einde van de variabele.

Slicing

Slicing is een techniek die eerder bij lijsten ter sprake is gekomen. Met vierkante haken kun je aangeven welke letters en karakters je precies uit een groter geheel wil plukken. Wil je alleen de eerste letter van een variabele? Dan gebruik je daarvoor [0]:

JSON-body

```
<#assign plaats = "Sintjohannesga">

{
  "action": "Substring: ${plaats[0]}"
}
```

Resultaat:



AUTOMATION,
Substring: S

Wil je de eerste tien letters? Dan gebruik je [0..10]:

JSON-body

```
<#assign plaats = "Sintjohannesga">

{
  "action": "Substring: ${plaats[0..10]}"
}
```

Resultaat:



AUTOMATION,
Substring: Sintjohanne

Een cheatsheet:

Variabele	Resultaat (vanuit waarde "Sintjohannesga")
\${plaats[0..1]}	Si
\${plaats[1..3]}	int
\${plaats[2..<5]}	ntj
\${plaats[2..5]}	ntjo
\${plaats[10..*3]}	esg
\${plaats[2..]}	ntjohannesga

De plus + operator


Een vervolg op slicing is de + operator, waarmee je waardes aan elkaar plakt:

JSON-body

```
<#assign plaats = "Sintjohannesga">

{
  "action": "Substring: ${plaats[0..2]} + plaats[4..5]}"
}
```

Resultaat:

 **AUTOMATION,**
Substring: Sinjo


De samengevoegde waardes zijn niet noodzakelijkerwijs altijd variabelen. Je kunt ook platte tekst erbij plaatsen, mits de tussen dubbele aanhalingstekens staan:

JSON-body

```
<#assign plaats = "Sintjohannesga">

{
  "action": "Substring: ${plaats[0..2]} + "ABCD" + plaats[4..5]}"
}
```

Resultaat:

 **AUTOMATION,**
Substring: SinABCDjo

Hetzelfde kun je overigens ook bereiken door de variabele `${plaats}` tweemaal aan te roepen:

JSON-body

```
<#assign plaats = "Sintjohannesga">

{
  "action": "Substring: ${plaats[0..2]} ABCD ${plaats[4..5]}"
}
```

De + operator heeft daarna ook een mathematische functie; je kunt er daadwerkelijk getallen mee optellen. Dat is echter buiten de scope van dit document.

Funcities combineren

Je kunt meerdere functies toepassen op één variabele. Je plaatst ze simpelweg achter elkaar, maar nog wel binnen de accolades {}.

JSON-body

```
<#assign info = "Pietje Bel">

{
  "action": "${info?ensure_starts_with("Naam: ")?ensure_ends_with(" (gebruiker)}"
}
```

Resultaat:

 **AUTOMATION,**
Naam: Pietje Bel (gebruiker)

Een ander voorbeeld:

JSON-body

```
<#assign info = "Naam: Jan, Achternaam: Visboer, Leeftijd: 30, Beroep: Koopman ">

{
  "action": "De voornaam is ${info?keep_after(": ")?keep_before(",")}"
}
```

Resultaat:

 **AUTOMATION,**
De voornaam is Jan

Ook slicing kun je met andere functies combineren. Wat zou het resultaat van onderstaande JSON-body zijn?

JSON-body

```
<#assign info = "Naam: Jan, Achternaam: Visboer, Leeftijd: 30, Beroep: Koopman ">

{
  "action": "De voornaam is ${info[8..20]?keep_after("a", "i")?keep_before_last("a", "i")}"
}
```

Tip: check het kopje "Derde argument" onder het hoofdstukje ?replace.

Uit de praktijk: data uit een ingediend formulier opzoeken

Onderstaande screenshot komt uit een formulier dat geautomatiseerd behandelaarsaccounts aanmaakt op basis van een emailadres:

Benodigde informatie

Emailadres *	<input type="text" value="jantje@internet.nl"/>
Behandelaarsgroep *	<input type="text" value="Servicedesk"/>

Met een GET-call kun je het verzoekveld uitlezen, bijvoorbeeld:

JSON-responsedata van stap getchange

```
Response body: {"results":[{"id":"fef66b7a-b9c3-404d-a2d9-fa6499eab18a","memoText":"Emailadres<br/>- jantje@internet.nl<br/><br/>Behandelaarsgroep<br/>- Servicedesk","entryDate":"2019-07-14T09:06:47+0000","operator":null,"person":null}]}
```

Het veld memoText bevat de ingevulde gegevens, en dat veld is hierboven gehighlight. De memoText haal je als volgt uit de responsebody:

```
<#assign verzoek = _responses.getchange.results[0].memoText>
```

Als we goed kijken, dan hebben we de data nodig na "Emailadres
- ", en voor "

Behandelaarsgroep
":

```
verzoek?keep_after("Emailadres<br/>- ")
```

```
"Emailadres<br/>- jantje@internet.nl<br/><br/>Behandelaarsgroep<br/>- Servicedesk"
```

```
verzoek?keep_before("<br/><br/>Behandelaarsgroep<br/>")
```

```
"Emailadres<br/>- jantje@internet.nl<br/><br/>Behandelaarsgroep<br/>- Servicedesk"
```

De twee functies ?keep_before en ?keep_after kun je vervolgens combineren. In twee stappen heb je dan het ingevulde emailadres:

```
<#assign verzoek = _responses.getchange.results[0].memoText>  
<#assign mailadres =  
verzoek?keep_after("Emailadres<br/>- ")?keep_before("<br/><br/>Behandelaarsgroep<br/>")>
```

Hiermee ontsluit je dus allerlei functionaliteit: je kunt nu geautomatiseerd acties uitvoeren op basis van data uit ingevulde formulieren. De mogelijkheden zijn eindeloos!

Regex (geavanceerd)

Tenslotte nog een korte opmerking over regex. Regex maakt het mogelijk om strings heel minutieus te vergelijken met patronen. Heb je geen ervaring met regex? Dan kun je dit kopje overslaan

Regex-patronen zijn op het eerste gezicht erg complex. Onderstaand is het regex-patroon voor alleen letters:

```
^([a-zA-Z]+)$
```

Met het regex-patroon beoordeelt een computer of een waarde geldig is (TRUE) of ongeldig is (FALSE):

Pietje	TRUE
Pietje Bell	FALSE (er zit een spatie in)
ABCD123	FALSE (er zitten cijfers in)
ABCdef	TRUE

Binnen actiereeksen

Je kunt regex gebruiken in de "Aangepaste voorwaarde", om te controleren op geldige waarden. Dat doe je met de `?matches`-functie:

Aangepaste voorwaarde

```
<#assign info = "ABCdef">  
<#if info??matches("^[a-zA-Z]+$")>true</#if>
```

```
<#assign info = "ABCdef">  
${info??matches("^[a-zA-Z]+$")}
```

Daarnaast kun je ook regex gebruiken met de `?replace`, `?keep_before`, `?keep_before_last`, `?keep_after`, `?keep_after_last` functies. Dat doe je door het extra argument "r" mee te geven. Voorbeeld:

JSON-body

```
<#assign info = "xxxxxxxAABB">  
  
{  
  "action": "Van ${info} naar ${info?replace("^(x)+", "y", "r")}"  
}
```

Resultaat:



AUTOMATION,

Van xxxxxxxAABB naar yAABB