Template Syntax

Grundlagen der Templating-Sprache

In diese Anleitung lernen Sie, wie Sie Ihre Daten in die resultierenden Ausgaben einbinden können.

Die gleiche Template-Syntax kann zudem in **Google Docs**, **LibreOffice**, **MS Word** und **HTML-Dokumenten** oder im **Plain Text** verwendet werden.
${if productName=Ultradox}

Sie können die Template-Syntax auch in allen ${productName}-Eingabefeldern verwenden, die blau oder grün hinterlegt sind. Dies ermöglicht es Ihnen, Formularfelder aus Ihrem Datum und mehr zu füllen.

Bei der Verwendung der Templating-Sprache in Dokumenten bleiben die Formatierungsmöglichkeiten, die Ihre Textverarbeitung bietet, erhalten.
${end}
Die Formatierungsmöglichkeiten von ${productName} ermöglichen es Ihnen, Text, Bilder, Karten, Tabellen oder Listen mit Daten aus beliebigen Datenquellen in Ihre Dokumente einzubinden.

Wir haben diese Core-Template-Engine auf github[extern] als Open Source zur Verfügung gestellt. Wenn Sie ein Java-Entwickler sind, können Sie es gerne in Ihren eigenen Projekten verwenden.

Variablen

Mit Variablen können Sie Daten in ein resultierende Dokument einbinden.
Variablen können ein einzelnes Wort, einen längeren Text, ein Datum, eine Zahl, ein Bild oder sogar eine Liste von Werten darstellen.
Nehmen wir an, Sie möchten einen personalisierten Brief erstellen. Ihr statisches Dokument kann die Ansage enthalten:

```
Hallo Daniel!
```

Nun wollen Sie den Namen Daniel durch eine Variable ersetzen, um personalisierte Briefe für verschiedene Kontakte generieren zu können. Das Ersetzen des Namens durch eine Variable ist einfach, geben Sie einfach ein

```
Hallo ${name}!
```

${productName} ist sehr wählerisch!
Variablennamen dürfen keine Sonderzeichen oder Leerzeichen enthalten!
Halten Sie Ihre Variablennamen einfach und verwenden Sie Kleinbuchstaben oder camelCase wie z.B. `nachName`, `telefonNummer` oder `eMail`.

Der Variablenname kann Unterstriche enthalten (z.B. `nach_name`), aber wir empfehlen, diese nicht zu verwenden.
Achten Sie darauf, dass Sie das Merge-Tag mit `\$\{` beginnen und mit `}` beenden.
Vertauschen Sie die geschweiften Klammern nicht mit eckigen Klammern oder ähnlichem und fügen Sie keine Leerzeichen hinzu.

Standardwerte

Wenn Sie personalisierte Briefe erstellen, die Sie an verschiedenen Kontakte senden möchten, kann es sein, dass Ihre Daten unvollständig sind und Sie entweder nur die E-Mail Adresse oder den Namen haben.
In diesem Fall könne Sie für die Variablen die nicht verfügbar sind, einen Standardwert angeben, der ausgegeben werden soll. Geben Sie diesen Standardwert in runden Klammern direkt nach dem Variablennamen an:

```
Hallo \$\{name(Kunde)}!
```

Wenn kein Standardwert festgelegt ist, steht in Ihrem Dokument lediglich:

```
Hallo Kunde!
```

Playground

Wenn Sie den Namen aus den Daten entfernen, wird stattdessen der Standardwert angezeigt.

```
tab:Template[interactive-jmte]
Hallo \${name(Kunde)}!
tab:Data[interactive-json]
{
  "name" : "Daniel"
}
tab:Output[transformed-plain]
```

Formatierung

${productName} bietet eine Reihe von *Renderern*, um die formatierung von Daten, Zahlen, Strings oder anderen Daten zu erleichtern.
Um einen Renderer zu verwenden, geben Sie einfach den Namen des Renderers ein und optional das Renderer-spezifische Format nach dem Variablennamen:

```
\$\{variable;renderer(format)}
```

Das `format` variiert je nach Renderer, aber in den meisten Fällen werden die Optionen aufgelistet und durch ein Semikolon getrennt.
Im Folgenden ein paar Beispiele, um zu sehen, wie das funktioniert:

Text in mehrere Zeilen aufteilen

Wenn Sie Text in mehrere Zeilen umbrechen möchten, können Sie im `text` Renderer die Option Wrap angeben.
Wenn die Variable `adresse` einen Text enthält, der so aussieht:

```
1600 Amphitheatre Parkway,Mountain View,CA 94043
```

können Sie die Angaben in verschiedene Zeilen aufteilen, z.B. mit der `wrap`-Option mit `,` als Trennzeichen:

```
\${contact.address;string(wrap=,)}
```

Dadurch wird die Adresse auf mehrere Zeilen aufgeteilt:

```
1600 Amphitheatre Parkway
Mountain View
CA 94043
```

Playground

Passen Sie die Adresse und das Trennzeichen an, um die resultierende Ausgabe zu ändern.

```
tab:Template[interactive-jmte]
Hi \$\{name},

Ihre Bestellung wird an folgende Adresse gesendet:

\$\{address;string(wrap=,)}
tab:Data[interactive-json]
{
  "name" : "Larry",
  "address" : "1600 Amphitheatre Parkway,Mountain View,CA 94043"
}
tab:Output[transformed-plain]
```

Umgang mit langen Texten

Wenn Ihre Variable sehr langen Text enthalten kann, können Sie die Länge des Textes mit dem Parameter `max` begrenzen.
Angenommen Sie haben eine Variable namens Beschreibung, die einen langen Texte wie diesen enthält:

```
Dies kann zu einem sehr langen Text werden, einem sehr langen Text,
der die Beschreibung eines Gegenstandes enthält
```

Sie können den Text nach 20 Zeichen mit dem Parameter `max` abschneiden:

```
\$\{description;string(max=20)}
```

Folgendes wird gedruckt:

```
Dies kann zu einem s
```

Um anzuzeigen, dass Text abgeschnitten wurde, können Sie die Option `ellipsis` angeben:

```
\$\{description;string(max=20;ellipsis=...)}
```

Die angegebenen Zeichen werden nach dem verkürzten Text angehängt, so dass Sie es jetzt erhalten:

```
Dies kann zu einem s...
```

Text in Groß- oder Kleinschreibung umwandeln

Verwenden Sie den `string` Renderer, wenn Sie Variablen nur in Groß- oder Kleinbuchstaben ausgeben möchten:

```
\$\{name;string(uppercase)} - \$\{name;string(lowercase)}
```

Wenn der Name `Daniel` ist, wird folgendes ausgegeben:

```
DANIEL - daniel
```

Kombinieren von Optionen

Natürlich können Sie alle erwähnten Optionen kombinieren.
Lassen Sie uns das Beispiel für den langen Text noch einmal zusammenfassen, um zu sehen, wie das funktioniert. Die Beschreibung der Variable enthält diesen längeren Text:

```
Dies kann zu einem sehr langen Text werden, einem sehr langen Text,
der die Beschreibung eines Gegenstandes enthält
```

Sie können nun die verschiedenen Optionen wie folgt kombinieren:

```
\$\{description;string(uppercase;wrap=, ;max=50;ellipsis=...)}
```

Dadurch wird die Variable `description` wie folgt ausgedruckt:

```
DIES KANN ZU EINEM SEHR LANGEN TEXT WERDEN
EINEM ...
```

Beachten Sie, dass in unserem Beispiel eine neue Zeile nur dann erstellt wird, wenn auf ein Komma ein Leerzeichen folgt.

Eine vollständige Beschreibung der Formatierungsmöglichkeiten des `string` Renderers finden Sie in der Renderer Reference.

Playground

Passen Sie die Parameter des `string` Renderer an, um zu sehen, wie sie sich auf das Ergebnis auswirken.

```
tab:Template[interactive-jmte]
Hi \$\{name},

Ihre Bestellung wird an folgende Adresse gesendet:

\$\{address;string(wrap=,)}

\$\{description;string(uppercase;wrap=, ;max=50;ellipsis=...)}
tab:Data[interactive-json]
{
  "name" : "Larry",
  "address" : "1600 Amphitheatre Parkway,Mountain View,CA 94043",
  "description" : "Dies kann zu einem sehr langen Text werden, einem sehr
langen Text, der die Beschreibung eines Gegenstandes enthält"
```

```
}
tab:Output[transformed-plain]
```

Dynamische Parameter

Angenommen, Sie generieren eine HTML E-Mail oder Website und möchten eine URL als Link darstellen. Sie können dafür den `link` Renderer verwenden und mit der Text-Option den anklickbaren Text angeben, der im erzeugten Dokument angezeigt werden soll:
`\${myurl;link(text=Click me)}`
Wenn der Text, den Sie anzeigen möchten, aus einer Variablen übernommen werden soll, anstatt ihn direkt in die Vorlage einzugeben, können Sie dynamische Parameter in den Renderer-Optionen verwenden:
`\${myurl;link(text=$mytext)}`
In diesem Beispiel wird der Text, der in Ihrem Dokument angezeigt wird, aus der Variablen `mytext` und der Link aus der Variable `myurl` übernommen.

Playground

Ändern Sie die Variablen `myurl` und `mytext` auf dem Tab **Daten**, um die Ausgabe des `link` Renderers zu ändern.

```
tab:Template[interactive-jmte]
<h1>Besuchen Sie uns auf der</h1>
<h2>\$\{myurl;link(text=$mytext;target=_blank)}</h2>
tab:Data[interactive-json]
{
  "myurl" : "https://cloud.withgoogle.com/next18/sf/",
  "mytext" : "Google Next"
}
tab:Output[transformed-html]
tab:Preview[preview-html]
```

Bedingungen

Bedingungen sind ein leistungsfähiges Feature, um anspruchsvolle Dokumente zu erstellen. Sie können zum Beispiel unterschiedlichen Text, verschiedene Fragen oder Absätze in Ihrem Dokument generieren, indem Sie Bedingungen festlegen und so die Dokumente auf Ihre Kunden oder bestimmte Benutzer individuell zuzuschneiden.
Die einfachste Bedingung überprüft, ob eine Variable gesetzt wurde und einen Wert hat, der nicht `false` ist.
Fügen wir zu unserem Dokument einen Text hinzu, für den Fall, dass der Benutzer männlich ist:
`\$\{if male}Guess what? You are male!\$\{end}`
Der Text zwischen dem `\$\{if ...}` und dem `\$\{end}` Tag wird nur gedruckt, wenn die “männlich” Variable gegeben ist und nicht `false` ist.
Wenn Sie einen Text hinzufügen möchten, wenn die Bedingung nicht erfüllt ist, können Sie das Tag `\$\{else}` verwenden:
`Sehr geehrter \$\{if male}Hr.\$\{else}Fr.\$\{end} \$\{familyName}`

Es mag ein wenig kompliziert aussehen, aber bei genauerer Betrachtung wird klar, was gemeint ist. Wenn der Kontakt männlich ist, wird es `Sehr geehrter Herr Schmidt` heißen, wenn der Kontakt nicht männlich ist, wird `Sehr geehrte Frau Schmidt` generiert.

Sie können überprüfen, ob eine Variable gesetzt ist, egal ob die Variable eine oder mehrere Werte enthält. Wenn Sie zum Beispiel einen Text oder eine Tabelle nur ausdrucken möchten, wenn das geladenes Blatt auch Textzeilen enthält, verwenden Sie einfach dieselbe Syntax:
`\$\{if sheet1}Das Arbeitsblatt sheet1 enthält mindestens eine Zeile\$\{end}`

${productName} erlaubt es Ihnen auch, **Ihre Variablen mit einem festen Wert zu vergleichen**. Nehmen wir an, Sie haben eine Variable namens `Land` und Sie möchten verschiedene Textzeilen auf Grundlage des Werts der Variable `Land` drucken, dann können Sie eine `if` Anweisung wie diese verwenden:

```
\$\{if country="DE"}Sie kommen aus Deutschland\$\{end}
\$\{if country="US"}Sie kommen aus den Vereinigten Staaten\$\{end}
\$\{if !country="DE"}Sie kommen nicht aus Deutschland\$\{end}
```

Benutzen Sie den Ausrufezeichen-Operator ! um Zeilen zu drucken, wenn die Variable nicht mit Ihrem angegebenen Text übereinstimmt

Playground

```
tab:Template[interactive-jmte]
\${if country="DE"}Sie kommen aus Deutschland\${end}
\${if country="US"}Sie kommen aus den Vereinigten Staaten\${end}
\${if !country="DE"}Sie kommen nicht aus Deutschland\${end}
tab:Data[interactive-json]
{
  "country" : "DE"
}
tab:Output[transformed-plain]
```

Bedingungen und Formatierung

Sie können alle Formatierungsoptionen verwenden, um erweiterte Bedingungen zu erstellen. Nehmen wir an, Sie haben eine Variable namens `email` mit dem Wert `username@gmail.com` und Sie wollen nur den Domain-Namen vergleichen, dann können Sie dafür den `string` Renderer nutzen:
`\$\{if email;string(fromAfterFirst=@)="gmail.com"}Sie verwenden GMail\$\{end}`
In diesem Beispiel gibt Ihnen der `string` Renderer den Teil der E-Mail nach dem `@` Zeichen aus.

Bitte seien Sie vorsichtig mit der Syntax bei der Kombination von Bedingungen und

Formatierung.

Bei der Formatierung Ihrer Variablen finden Sie eine Reihe von Optionen, die entweder als `true` (wahr) oder `false` (falsch) ausgegeben werden. Diese Optionen sind dadurch für Bedingungen mit der `if`-Syntax geeignet.
Wenn Sie z. B. prüfen möchten, ob die Variable `alter` eine Zahl größer als 21 enthält, können Sie diese Anweisung verwenden:

```
\$\{if alter;number(gt=21)}Sie sind in dem gesetzliche
vorgeschriebenen Alter\$\{end}
```

Wenn Sie einen Variablennamen mit Wert `Oliver Angermann` haben und prüfen möchten, ob der Nachname mit einem A beginnt, können Sie die Formatierungsoptionen wie folgt kombinieren:

```
\$\{if name;string(fromAfterLast= ;startsWith=A)}Das letztes Wort
beginnt mit einem A\$\{end}
```

Wenn Sie die Variable `termin` mit dem Wert `2018/04/03` haben und wollen überprüfen, ob es im nächsten Jahr ist, können Sie es mit dem Ausgabeformat `yyyy` kombinieren, um das Jahr mit dem `addYear` und der `equals` Option zu vergleichen:

```
\$\{if termin;date(op=YYYY;addYear=-1;equals=$now)}Freigabe ist
nächstes Jahr!\$\{end}
```

Verschachtelte Bedingungen

Durch das Verschachteln von Bedingungen können Sie `AND` und `OR` Operatoren emulieren.
Versuchen wir folgende Aufgabe zu lösen: Sie betreiben Ihr Unternehmen in der EU und möchten eine Rechnung für einen europäischen Kunden erstellen, so dass Sie nur die Mehrwertsteuer hinzufügen müssen, wenn der Kunde Privatkunden ist.
Sie können das **Reverse Charge** Verfahren verwenden, wenn der Kunde eine Firma ist, die durch eine gültige Umsatzsteuer-Identifikationsnummer gekennzeichnet ist.
Nehmen wir an, Sie haben ${productName} in Ihren eigenen Onlineshop integriert und haben die Variablen `vatId` und `euCustomer` zur Verfügung gestellt:

```
tab:Template[interactive-jmte]
\${if euCustomer}
Sie sind ein europäischer Kunde.
\${if vatId}
Verifizierte Umsatzsteuer-ID ist \${vatId}, Wir weisen Sie auf Ihre
Steuerschuldnerschaft ("Reverse-Charge"), gemäß § 13b UStG hin.
\$\{else}
${vatRate;number(op=0.0)}% Inklusive Mehrwertsteuer für Privatkunden.
\${end}
\${else}
Sie sind ein internationaler Kunde außerhalb der EU.
\${end}
tab:Data[interactive-json]
{
  "euCustomer" : true,
```

```
  "vatId" : "DE1234567890",
  "vatRate" : 21
}
tab:Output[transformed-plain]
```

Listen

Variablen können auch eine Liste mit definierten Werten enthalten.
Wenn Sie zB auf einen Kontakt mit mehreren E-Mail Adressen zugreifen, stellt ${productName} eine Variable zur Verfügung, die eine Liste aller E-Mail Adressen enthält.
Bei der Arbeit mit Tabellenkalkulationen bietet ${productName} eine Variable, die die Liste der geladenen Zeilen mit benannten Werten für jede Spalte enthält.

Einfache Variablen und Listen-Variablen werden durch das winzige Icon hinter dem Variablennamen unterschieden: eine einzelne Zeile zeigt an, dass dies eine einfache Variable mit nur einem einzigen Wert ist, drei Zeilen zeigen an, dass diese Variable eine Liste von Werten darstellt.
Sie können die Listen-Variablen auch durch die zwei Klammern hinter dem Variablennamen identifizieren. Z.B. die Variablen `contacts[].givenName` und `contacts[].photo` beim Laden mehrerer Kontakte.

Soll der Titel des ersten Kontaktes ausgedruckt werden, schreiben Sie`\$\{contacts[0].title}`, um den Titel des dritten Kontakts zu drucken `\$\{contacts[2].title}` und um auf den letzten Kontakt in der Liste zuzugreifen `\$\{contacts[last].title}`. Diese Funktion kann auch praktisch sein, wenn Sie beispielsweise auf die letzte Zeile eines Arbeitsblatts zugreifen möchten.
`\$\{contacts[].length}` druckt die Anzahl der geladenen Kontakte.

```
tab:Template[interactive-jmte]
Anzahl der Kontakte: \${contacts.length}

Name des ersten Kontakts: \${contacts[0].name}
Name des zweiten Kontakts: \${contacts[1].name}
Name des letzten Kontakts: \${contacts[last].name}
tab:Data[interactive-json]
{
  "contacts" : [
    {
      "name" : "Daniel"
    },
    {
      "name" : "Olli"
    },
    {
      "name" : "Esther"
    },
    {
      "name" : "Alex"
    }
```

```
  ]
}
tab:Output[transformed-plain]
```

Schleifen

${productName} ermöglicht es Ihnen auch, Werte zu iterieren, indem Sie das Schlüsselwort `foreach` verwenden.
Nehmen wir an, wir haben eine Liste von Kontakten mit den Feldern `name` und `email`.
Das erste Argument in der `foreach` Anweisung ist die Variable, die die Liste enthält, in unserem Beispiel ist es die Variable `contacts`, die die Liste der Kontakte enthält.
Das zweite Argument in der foreach-Anweisung ist der Iterator, der uns helfen wird, auf die Felder jedes Kontakts in der Schleife zuzugreifen.
Im Beispiel werden wir den `contact` als Namen für den Iterator wählen, aber Sie können stattdessen einen beliebigen Namen wählen - stellen Sie einfach sicher, dass Sie diesen Namen beim Zugriff auf die Felder verwenden.

```
tab:Template[interactive-jmte]
<ul>
\$\{foreach contacts contact}
<li>\${contact.name} - \${contact.email}</li>
\$\{end}
</ul>
tab:Data[interactive-json]
{
  "contacts" : [
    {
      "name" : "Daniel",
      "email" : "d@floreysoft.net"
    },
    {
      "name" : "Olli",
      "email" : "o@floreysoft.net"
    },
    {
      "name" : "Esther",
      "email" : "e@floreysoft.net"
    },
    {
      "name" : "Alex",
      "email" : "a@floreysoft.net"
    }
  ]
}
tab:Output[transformed-html]
tab:Preview[preview-html]
```

Sie können Bedingungen in Schleifen verwenden und sogar Schleifen verschachteln, bei denen Sie es mit einer Liste von Listen zu tun haben.

Was ist, wenn wir alle E-Mail Adressen in einer einzigen Zeile, getrennt durch ein Komma, ausdrucken wollen? Wenn wir einen solchen Ansatz verfolgen:

```
\${foreach contacts contact}\${contact.name}, \${end}
```

haben wir dieses Ergebnis:

```
Daniel, Olli, Esther, Alex,
```

Das sieht gut aus, außer, dass wir auch ein Komma am Ende der Liste haben.
Wie können wir das vermeiden?

Schleifen mit Trennzeichen

${productName} bietet eine spezielle Funktion an, mit der Sie Schleifen mit einem Trennzeichen erstellen können:

```
\${foreach contacts contact , }\${contact.name}\${end}
```

So können Sie im `foreach` Konstrukt ein Trennzeichen eingeben. Sie können ein einzelnes oder mehrere Zeichen als Trennzeichen verwenden.
In unserem Fall verwenden wir ein Komma gefolgt von einem Leerzeichen. Das Ergebnis sieht so aus:

```
Daniel, Olli, Esther, Alex
```

Das ist, was wir gesucht haben!

Werte umschließen

Geben Sie Zeichen an, die beim rendern einer Variablen vorangestellt oder angehängt werden:

```
\$\{<h1>,title,</h1>}
```

Wenn die Variable `titel` “Hallo Welt” enthalten würde, würde es zu diesem Ergebnis führen:

```
<h1>Hallo Welt</h1>
```

Dies ist großartig, wenn Sie HTML-Dokumente erstellen, kann aber auch in anderen Fällen nützlich sein.
Zum Beispiel, wenn Sie für Benutzer ohne Vorgabe eines bestimmten Namens folgendes drucken wollen:

```
Hi!
```

und für Benutzer mit angegebenem Namen:

```
Hi Daniel!
```

Wenn Sie einfach die Anweisung `Hi \${if name}!` verwenden und der Name nicht angegeben ist, wirst es mit `Hi !` enden, das ein unerwünschtes Leerzeichen enthält. Wenn Sie den Platz entfernen, erhalten Sie `HiDaniel!`, wenn der Name gesetzt ist - was noch schlimmer ist.
Sie können das Problem lösen, indem Sie eine Bedingung verwenden:

```
Hi\${if name} \${name}\${else}\${end}!
```

Dieser Ausdruck würde prüfen, ob der Name gesetzt ist und nur dann ein Leerzeichen hinzufügen, wenn der Name vorhanden ist.
Dies ist eine ziemlich lange Aussage, nur um das Leerzeichen loszuwerden und kann durch die Verwendung der folgenden Abkürzung vereinfacht werden:

```
Hi\$\{ ,name,}!
```

Für Experten

Wenn Sie mit Templating Sprachen vertraut sind, zeigen wir Ihnen hier ein Beispiel mit einer voll funktionsfähigen HTML-Vorlage und Daten, die mit einem Webhook abgerufen wurden:

```
tab:Template[interactive-jmte]
<!-- Namen wird hervorgehoben und Standardwert bereitgestellt-->
<p>Hallo \${<b>,name(dear customer),</b>},</p>

<!-- Datum und Uhrzeit in Kurzform ausgeben -->
<p>wir haben Ihre Bestellung am \${date;date(op=short;ol=de)} um
\${date;time(op=short;ol=de)} erhalten.</p>

<!-- Drucken Sie alle Autoren getrennt durch Komma -->
<p>Sie haben Bücher von folgenden Autoren bestellt: \${foreach
authors author , }\${author.name}\${end}.</p>

<!-- Bereich für alle Autoren... -->
\${foreach authors author}
<p><b>\${author.name}</b></p>
<table>
<tr><th width="200"
style="text-align:left">Titel</th><th>Ausgabe</th></tr>

<!-- ... und alle Bücher dieses Autors anzeigen -->
\${foreach author.books book}
<tr><td width="200">\${book.title}</td><td
style="text-align:left">\${book.published}</td></tr>
\${end}
</table>
\${end}
${end}
tab:Data[interactive-json]
{
  "name" : "Daniel",
  "date" : "2018-02-25T13:15",
  "authors" : [
    {
      "name" : "Arnon Grunberg",
```

```
      "books" : [
        {
          "title" : "Mit Haut und Haaren",
          "published" : "2012"
        },
        {
          "title" : "Tirza",
          "published" : "2006"
        }
      ]
    },
    {
      "name" : "Fjodor Michailowitsch Dostojewski",
      "books" : [
        {
          "title" : "Schuld und Sühne",
          "published" : "1864"
        },
        {
          "title" : "Die Brüder Karamasow",
          "published" : "1877"
        }
      ]
    }
  ]
}
tab:HTML[transformed-html]
tab:Preview[preview-html]
```

Erfahren Sie mehr

Weitere Informationen über die Templating-Sprache finden Sie in folgenden Anleitungen.
${if productName=Doxey}

Dokumentvorlagen | Internationalisierung | Renderer Referenz

${else}

Dokumentvorlagen | Internationalisierung | Renderer Referenz

${end}