ICT Berufsbildung
Formation professionnelle
Formazione professionale

ICTskills 2023 | Skill 09

mustache manual excerpt

Version 1.0

Version: 1.0
Authors: Thomas Kehl, Ueli Niederer

ICT-Berufsbildung Schweiz
Waisenhausplatz 14, CH-3011 Bern
Telefon +41 58 360 55 50
info@ict-berufsbildung.ch
www.ict-berufsbildung.ch

mustache manual excerpt

Table of contents

1 Introduction

mustache is a popular logic-less template engine [1] featuring a simple syntax allowing users to render dynamic data into predefined templates. A multitude of libraries exist, implementing the template engine for various ecosystems such as C#/.NET, Java, C++, JavaScript, etc. Feel free to use a mustache library if available in your ecosystem.

The following sections are an excerpt out of mustache's documentation [2] outlining the minimal functionality expected to be working within your assignment in case you decided to implement the template engine yourself.

2 Synopsis

A typical Mustache template:

```
Hello {{name}}
You have just won {{value}} dollars!
{{#in_ca}}
Well, {{taxed_value}} dollars, after taxes.
{{/in_ca}}
```

Given the following hash:

```
{
  "name": "Chris",
  "value": 10000,
  "taxed_value": 10000 - (10000 * 0.4),
  "in_ca": true
}
```

Will produce the following:

```
Hello Chris
You have just won 10000 dollars!
Well, 6000.0 dollars, after taxes.
```

3 Description

Mustache can be used for HTML, config files, source code - anything. It works by expanding tags in a template using values provided in a hash or object.

We call it "logic-less" because there are no if statements, else clauses, or for loops. Instead there are only tags. Some tags are replaced with a value, some nothing, and others a series of values. This document explains the different types of Mustache tags.

4 Tag Types

4.1 Variables

The most basic tag type is the variable. A **{{name}}** tag in a basic template will try to find the **name** key in the current context. If there is no **name** key, the parent contexts will be checked recursively. If the top context is reached and the **name** key is still not found, nothing will be rendered.

All variables are HTML escaped by default. If you want to return unescaped HTML, use the triple mustache: **{{{name}}}**.

By default a variable "miss" returns an empty string. This can usually be configured in your Mustache library. The Ruby version of Mustache supports raising an exception in this situation, for instance.

Template:

```
* {{name}}
* {{age}}
* {{company}}
* {{{company}}}
```

Hash:

```
{
  "name": "Chris",
  "company": "<b>GitHub</b>"
}
```

Output:

```
* Chris
*
* &lt;b&gt;GitHub&lt;/b&gt;
* <b>GitHub</b>
```

4.2 Sections

Sections render blocks of text one or more times, depending on the value of the key in the current context.

A section begins with a pound and ends with a slash. That is, **{{#person}}** begins a "person" section while **{{/person}}** ends it.

The behavior of the section is determined by the value of the key.

False Values or Empty Lists

If the **person** key exists and has a value of false or an empty list, the HTML between the pound and slash will not be displayed.

Template:

```
Shown.
{{#person}}
  Never shown!
{{/person}}
```

Hash:

```
{
  "person": false
}
```

Output:

```
Shown.
```

Non-empty-Lists

If the **person** key exists and has a non-false value, the HTML between the pound and slash will be rendered and displayed one or more times.

When the value is a non-empty list, the text in the block will be displayed once for each item in the list. The context of the block will be set to the current item for each iteration. In this way we can loop over collections.

Template:

```
{{#repo}}
  <b>{{name}}</b>
{{/repo}}
```

Hash:

```
{
  "repo": [
    { "name": "resque" },
    { "name": "hub" },
    { "name": "rip" }
  ]
}
```

Output:

```
<b>resque</b>
<b>hub</b>
<b>rip</b>
```

5 References

[1] C. Wanstrath, «{{ mustache }},» 19 01 2021. [Online]. Available: https://mustache.github.io/. [Zugriff am 19 07 2023].

[2] C. Wanstrath, «mustache manual,» 04 2014. [Online]. Available: https://mustache.github.io/mustache.5.html. [Zugriff am 19 07 2023].