Advanced-Template-Markup-Format (ATMF)

Cultural Made Easy

Version 1

By Dimitar Atanasov

Dec 7, 2020

This document describes the concept of the Advanced Template Markup Format (ATMF) syntax and technological relations. The paper is platform and framework independent.

Contents

What it is?

ATMF is HTML based format which is specially designed to simplify linking between the front and back end. It provides high availability of server-side data at template level.

Why is needed?

Building of medium-high complex localized web solutions can be real time and effort consumer, especially when it comes to providing front-end templates for UI manipulation without changing the back-end. Adding multi-cultural support to the apps usually takes a lot of efforts and additional steps at development stage which make it harder to build and scale. ATMF is designed to skip all the additional "work to be done" and allow the developers to start creating localized templates with highly available back-end data immediately with ease.

What it looks like?

The template:

```
<html>
  <head>
    <title>{@header.title}</title>
  </head>
  <body>
    <h1>{@header.helloWorld}</h1>
    <p>{@page.todayDate} {/fdate $date}</p>
    <p>{@page.visitorNum $count}</p>
    <p>{@page.others $othersCount}</p>
  </body>
</html>
```

The output:

```
<html>
  <head>
    <title>Hello World</title>
  </head>
  <body>
    <h1>Hello World</h1>
    <p>Today is 2020-12-04</p>
    <p>You are visitor number 7</p>
    <p>6 other visitors came before you.</p>
  </body>
</html>
```

The translation:

```
en/page.json
{
  todayDate: "Today is",
  vistorNum: "You are visitor number $0"
  others: [
    "$0 more visitors came before you",
    "One more visitor came before you"
  ]
```

What it takes?

It just needs server-side processor and it's ready to go. Its compatible with any web framework that uses HTML as PHP, Python, .NET, Node.Js, etc.

Go deeper…

ATMF uses four component **`SLVE(#@$/)`** syntax system.

```
# System Function
@ Language Resource
$ Variable
/ Extension
```

Where each action or resource is defined by its special char `#@$/` making template building super-fast and easy.

System functions

These include: `#if` `#else` `#endif` `#use` `#template`

```
<html>
  <head>
    <title>{@header.title}</title>
  </head>
  <body>
    <!-- Nested templates - handle master/page views -->
    <div>{@template header}</div>
    <div>{@template page}</div>
    <div>{@template footer}</div>
  </body>
</html>
```

@ Language resources

The interpreter automatically discovers language resources target with @. Translations are organized in files in special folder order:

```
culture/
|__en-US/
   |__.culture.json
   |__page.json
   |__header.json
   |__settings/
      |__profile.json
|__bg-BG/
   |__.culture.json
   |__page.json
   |__header.json
   |__settings/
      |__profile.json
```

Each language resource key is consisting of @path/file.key, like @settings/profile.name. It can be also combined with the #use for additional simplifying.

```
{@use lang settings}
{@profile.name}
```

OR

```
{@use lang settings as p}
{@p.name}
```

The language resources can be also combined with $variables like: {@profile.name $fullName}. This will pass $fullName value to @profile.name language resource. Multiple values in the translations can be captured by index variables $0, $1, $2 etc..

```
<!-- Translation resource -->
theFox: "The $1 fox is made $0 steps"

<!-- Usage -->
{@page.theFox $steps $color}
```

ATMF is easily handling the plural nouns.

```
<!-- Translation resource -->
theFox: [
          "The $1 fox is made $0 steps",
          "The $1 fox is made one step"
]

<!-- Usage -->
{@page.theFox $steps $color}
```

The above will automatically decide whether single or plural noun sentence should be used based on the first parameter. If it's number and equal to 1 then it will be single noun sentence. Anything else will evaluate to plural.

```
<!-- Usage -->
{@page.theFox 1 red}   <!-- Output: The red fox made one step -->
{@page.theFox 8 blue}  <!-- Output: The blue fox made 8 steps -->
{@page.theFox 0 green} <!-- Output: The green fox made 0 steps -->
```

$ Variables

They are assigned from the back-end and can be used separately or in combinations with other operators.

```
{$totalCount}
{@page.hello $fullName}
```

/ Extensions

These are third-party functions for processing the template data. Same as the system functions they accept parameters.

```
{/myformat $toralCount @page.steps}
{@page.steps {/myformat $totalCount}}
```