Template Engines in Rust

Sebastian Neubauer

February 13, 2020

What is a Template Engine?

- Create text as output, e.g. website, generated code, e-mails
- Mix sections of output text and code
- Code generates dynamic text

Template Engines in Rust

Tera

Handlebars

Askama

Maud

t4rust

Liquid

Example

Tera

```
Hi {{ mail }},

we have a nice offer for u. You only have time until tomorrow to
accept the {{ amount }} money I will donate to you.
<a href="http://tota.lly.leg.it/website.php/{{ uuid }}">
Click here to receive {{ amount }} for free!!1!</a>

{% if confirmed %}
We know you read this!
{% endif %}

Yours sincrly,
Mr. {{ name }}
```

It gets harder

Tera

- Print something 100 times
- `while`-loops
- Print a number in hex

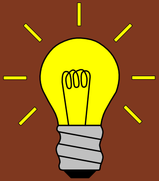
It gets harder

Tera

- Print something 100 times
- `while`-loops
- Print a number in hex

…not that easy and intuitive.
⇒ There must be an easier way!

Idea

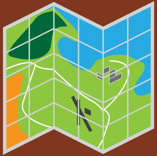


We use Rust anyway, why not in templates too?

Code to generate

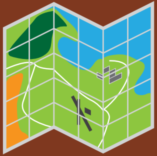
```rust
fn fmt(&self, f: &mut fmt::Formatter) -> fmt::Result {











    Ok(())
}
```

Code to generate



```
fn fmt(&self, f: &mut fmt::Formatter) -> fmt::Result {
    write!(f, "Hi ")?;
    write!(f, "{}", self.mail)?;
    write!(f, ",

we have a nice offer for u.")?;
    // ...





    Ok(())
}
```

Code to generate



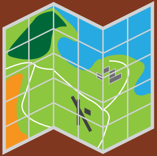
```
fn fmt(&self, f: &mut fmt::Formatter) -> fmt::Result {
    write!(f, "Hi ")?;
    write!(f, "{}", self.mail)?;
    write!(f, ",

we have a nice offer for u.")?;
    // ...

    if self.confirmed {
        write!(f, "We know you read this!")?;
    }
    // ...
    Ok(())
}
```

Code to generate

```

    write!(f, "Hi ")?;
    write!(f, "{}", self.mail)?;
    write!(f, ",

we have a nice offer for u.")?;


    if self.confirmed {
        write!(f, "We know you read this!")?;
    }



```

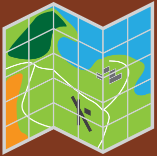
Code to generate

```

    write!(f, "Hi ")?;
    write!(f, "{}", self.mail)?;
    write!(f, ",

we have a nice offer for u.")?;


    if self.confirmed {
        write!(f, "We know you read this!")?;
    }



```

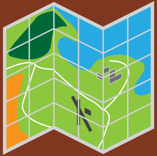


Code to generate

```
                 Hi
                  <#= self.mail #>
                 ,

we have a nice offer for u.


 <# if self.confirmed { #>
                     We know you read this!
 <# } #>



```



Code to generate



```
<#@ template cleanws="true" #>
Hi <#= self.mail #>,

we have a nice offer for u. You only have time until tomorrow to
accept the <#= self.amount #> money I will donate to you.
<a href="http://tota.lly.leg.it/website.php/<#= self.uuid #>">
Click here to receive <#= self.amount #> for free!!1!</a>

<# if self.confirmed { #>
We know you read this!
<# } #>

Yours sincrly,
Mr. <#= self.name #>
```

Usage

t4rust

```rust
use t4rust_derive::Template;

#[derive(Template)]
#[TemplatePath = "mail_template.tt"]
struct Mail {
    mail: String,
    amount: u64,
    uuid: String,
    confirmed: bool,
    name: String,
}

fn main() {
    let mail = Mail {
        mail: "info@customer.com".into(),
        amount: 1_000_000,
        uuid: "2349753982734".into(),
        confirmed: true,
        name: "Sandman".into(),
    };
    print!("{}", mail);
}
```

Comparison

t4rust

Advantages	Disadvantages
Do everything you want	No untrusted templates
Same language everywhere	No auto-escaping
Compile-time errors	Needs recompilation
Simple	No good error messages yet

You know something we don't?

We await your talk!

Writing Functions

t4rust

```
<#
fn fun(_fmt: &mut fmt::Formatter, old: bool) -> fmt::Result {
    if !old { #>
A brand new world!
    <# } else { #>
The stable and working version.
    <# }

    Ok(())
}
#>

Here it was: <# fun(_fmt, true) #>
And here it comes: <# fun(_fmt, false) #>
```