Template Specification[¶](#template-specification)

Templates abide by the following rules:

* Templates are text files with sections marked off called variables
* Variables will be marked for replacement at render time.
* Variables are surrounded with double curly braces, *{{* and *}}*.
* Variables can specify a type: *name*::*type*
* Primitive Types
  + string (default)
  + text (for strings with newlines and escape characters)
  + number
  + integer
  + boolean
  + array

Example[¶](#example)

The following is an example of a simple FAST template that will render an AS3 declaration:

{

"class": "ADC",

"schemaVersion": "3.11.0",

"{{tenant\_name}}": {

"class": "Tenant",

"{{application\_name}}": {

"class": "Application",

"template": "http",

"serviceMain": {

"class": "Service\_HTTP",

"virtualAddresses": ["{{virtual\_address}}"],

"pool": "web\_pool\_{{port}}",

},

"web\_pool\_{{port}}": {

"class": "Pool",

"monitors": [

"http"

],

"members": [

{

"servicePort": {{port::integer}},

"serverAddresses": {{server\_addresses::array}}

}

]

}

}

}

}

}

In the example template, we have some variables: tenant\_name, application\_name, virtual\_address, port, and server\_addreses. Some have annotations, like *port::integer*. The *integer* annotation signifies the value of *port* must be an integer.

Variables may be used in multiple places, if a variable is annotated somewhere in the file, an unannotated version of that variable will respect the annotation.

From the variables, a schema is generated. This schema describes the parameters that must be provided to render the template. These parameters will show up in the form representation of the template in the GUI.

The following schema will get auto-generated from the example:

{

"properties": {

"tenant\_name" : {

"type": "string"

},

"application\_name" : {

"type": "string"

},

"virtual\_address" : {

"type": "string"

},

"server\_addresses" : {

"type": "array"

},

"port" : {

"type": "integer"

},

}

}

This example ‘view’ passes validation using the schema:

{

"tenant\_name" : "myTenant",

"application\_name" : "simple\_http\_1",

"virtual\_address" : "10.0.0.1",

"server\_addresses" : [ "10.0.1.1", "10.0.2.2" ],

"port" : 80

}

This information is collected in the form UI, and compiled into a parameter object like the example. The information is passed along to the template renderer, and the variable names are replaced with their parameter values.

The final declaration is generated by providing the previous view with the provided template:

{

"class": "ADC",

"schemaVersion": "3.11.0",

"myTenant": {

"class": "Tenant",

"simple\_http\_1": {

"class": "Application",

"template": "http",

"serviceMain": {

"class": "Service\_HTTP",

"virtualAddresses": ["10.0.0.1"],

"pool": "web\_pool\_80",

},

"web\_pool\_80": {

"class": "Pool",

"monitors": [

"http"

],

"members": [

{

"servicePort": 80,

"serverAddresses": [ "10.0.1.1", "10.0.2.2" ]

}

]

}

}

}

}

}