# {{ project.name }}

This template is an example of how Jinja2 variables can be used in a Word template to dynamically drop-in information and generate tables. Use this template to generate a report to see how it looks.

## The Report Data

Generate a JSON report for your project to see the data accessible from within your templates. You can access the values just like you would in a language like Python–i.e., you will reference the key and use periods to access nested values. For example, here we can construct a string from the client and project keys:

{{ client.name }} ({{ client.short\_name }}) {{ project.type }} (generated on {{ report\_date }})

Some of this data is chopped-up into smaller bits to make it easy to reassemble in different ways. For example, the project’s start and end dates are accessible as pre-formatted dates based on your date format configuration:

{{ project.start\_date }} – {{ project.end\_date }}

Perhaps you want to reference only pieces (e.g., day, month, and year) of these dates to present them in different ways. Here is one option for dynamically assembling a date range:

{% if project.start\_year == project.end\_year %}{% if project.start\_month == project.end\_month %}{{ project.start\_month }} {{ project.start\_day }}–{{ project.end\_day }}, {{ project.end\_year }}{% else %}{{ project.start\_day }} {{ project.start\_month }} to {{ project.end\_day }} {{ project.end\_month }} {{ project.end\_year }}{% endif %}{% else %}{{ project.start\_day }} {{ project.start\_month }} {{ project.end\_year }} to {{ project.end\_day }} {{ project.end\_month }} {{ project.end\_year }}{% endif %}

That looks kind of complicated, but it outputs several different date formats based on the months and years involved. You can add newlines to break-up the statements to make them easier to read. We want to write Jinja2 statements in one line in our final template, so we don’t end up with blank lines and extra whitespace when the Jinja2 expressions are removed during rendering.

## Plain Text and Rich Text

In Ghostwriter, you edit some content in a WYSIWYG editor where you can apply styles. The formatted text is stored as HTML, so inserting that text will include HTML tags. You can use the strip\_html filter to remove the tags, but you probably want to include your formatting. To do that, you can reference the RichText object version of the variable (denoted by the \_rt suffix).

For example, here is the project description in both forms:

**Raw:**

{{ project.note }}

**Stripped HTML:**

{{ project.note|strip\_html }}

**RichText:**

{{p project.note\_rt }}

Check the wiki for more details: <https://www.ghostwriter.wiki/features/reporting/report-templates/word-template-variables>

## Making Tables & Using Math

There is a lot you can do with Jinja2. Here we will look at some common things you are likely to want to do in a report, make tables and calculate values.

You can pull in some pre-calculated values from the totals key and perform calculations:

Table 1 – Assessment Results

|  |  |  |
| --- | --- | --- |
| Value | Total | Math |
| **Objectives** | {{ totals.objectives }} | -- |
| **Completed Objectives** | {{ totals.objectives\_completed }} | {% if totals.objectives > 0 %}{{ totals.objectives\_completed / totals.objectives \* 100 }}{% else %}100{% endif %}% |
| **Findings** | {{ totals.findings }} | -- |
| **Critical Findings** | {{ totals.findings\_critical }} | {% if totals.findings\_critical > 0 %}{{ totals.findings\_critical / totals.findings \* 100 }}{% else %}100{% endif %}% |
| **High Findings** | {{ totals.findings\_high }} | {% if totals.findings\_high > 0 %}{{ totals.findings\_high / totals.findings \* 100 }}{% else %}0{% endif %}% |
| **Medium Findings** | {{ totals.findings\_medium }} | {% if totals.findings\_medium > 0 %}{{ totals.findings\_medium / totals.findings \* 100 }}{% else %}0{% endif %}% |
| **Low Findings** | {{ totals.findings\_low }} | {% if totals.findings\_low > 0 %}{{ totals.findings\_low / totals.findings \* 100 }}{% else %}0{% endif %}% |
| **Info Findings** | {{ totals.findings\_info }} | {% if totals.findings\_info > 0 %}{{ totals.findings\_info / totals.findings \* 100 }}{% else %}0{% endif %}% |
| **Team Members** | {{ totals.team }} | -- |
| **Targeted Hosts** | {{ totals.targets }} | -- |
| **Scope** | {{ totals.scope }} | -- |

Be mindful of performing math like dividing a value that could be zero. In the above examples, this template would fail rendering with an error if a value like total.objectives was zero and the block did not have if totals.objectives > 0.

You will see an error like this:

Word document generation failed because the selected template has Jinja2 code that attempts to divide by zero

Tip: Before performing math, check if the number is greater than zero

You might want to get a value that isn’t pre-calculated. You can count something using the Jinja2 length filter. For example, this pre-calculated value for the total number of findings is equal to the output of this filter: {{ totals.findings }} == {{ findings|length }}

You can also create tables with loops. Note the use of a special prefix, {tr, to start the blocks. These table rows will not appear in your final document.

Table 2 – {{ client.name }} Points of Contact

|  |  |  |
| --- | --- | --- |
| Name | Role | Email |
| {%tr for poc in client.contacts %} | | |
| {{ poc.name }} | {{ poc.job\_title }} | {{ poc.email }} |
| {{p poc.note\_rt }} | | |
| {%tr endfor %} | | |

Table 3 – {{ company.name }} Points of Contact

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Role | Email | Phone |
| {%tr for member in team %} | | | |
| {{ member.name }} | {{ member.role }} | {{ member.email }} | {{ member.phone }} |
| {{p member.note\_rt }} | | | |
| {%tr endfor %} | | | |

Table 4 – Domain Names Used for Assessment Activities

|  |  |  |
| --- | --- | --- |
| Domain Name | Role | Note |
| {%tr for domain in infrastructure.domains %} | | |
| {{ domain.domain }} | {{ domain.activity }} | {{p domain.note\_rt }} |
| {%tr endfor %} | | |

Table 5 – Servers Used for Assessment Activities

|  |  |  |  |
| --- | --- | --- | --- |
| IP Address | Purpose | Role |  |
| {%tr for server in infrastructure.servers %} | | | |
| {{ server.ip\_address }} | {{ server.activity }} | {{ server.role }} | {{p server.note\_rt }} |
| {%tr endfor %} | | | |
| {%tr for server in infrastructure.cloud %} | | | |
| {{ server.ip\_address }} | {{ server.activity }} | {{ server.role }} | {{p server.note\_rt }} |
| {%tr endfor %} | | | |

Here is an example of some light table formatting that is possible via a Jinja2 expression. The cellbg expression will set a background color for the cell. In this case, we set the background color of the *Severity* column to match the color assigned to each finding’s severity:

Table 6 – Summary of Findings

|  |  |
| --- | --- |
| Finding | Severity |
| {%tr for finding in findings %} | |
| {{ finding.title }} | {% cellbg finding.severity\_color %}{{ finding.severity }} |
| {%tr endfor %} | |

## Using Filters

Filters have already been used in previous sections, but let’s take a closer look at some examples.

There are numerous Jinja2 filters available within templates. Ghostwriter also has some custom filters (see the wiki). We can recreate the findings table with the filter\_severity filter.

This version only includes findings that are Critical or High severity:

|  |  |
| --- | --- |
| Finding | Severity |
| {%tr for finding in findings|filter\_severity([“Critical”, “High”]) %} | |
| {{ finding.title }} | {% cellbg finding.severity\_color %}{{ finding.severity }} |
| {%tr endfor %} | |

Depending on your findings, that table could appear empty (just the header row and nothing else). That’s no good.

You can check for conditions before including something in your report. Let’s say you want to use the filter\_type filter to make a table of only certain types of findings, but you might not always need it. You can check if those findings exist before including the table: {% if findings|filter\_type([“Network”, “Web”]) %}

|  |  |
| --- | --- |
| Finding | Severity |
| {%tr for finding in findings|filter\_type([“Network”, “Web”]) %} | |
| {{ finding.title }} | {% cellbg finding.severity\_color %}{{ finding.severity }} |
| {%tr endfor %} | |

{% else %}Nothing to see here! {% endif %}Learn more about Jinja2’s built-in filters: <https://jinja.palletsprojects.com/en/3.0.x/templates/#list-of-builtin-filters>

## Advanced Variable Usage

You may have noticed the previous example ran the filter twice–once to check the condition and again to perform the loop. There may be times you want to check a condition, calculate a sum, or generate some other value and then use it more than once. Rather than recreating the value multiple times, you can use the with block.

This block checks the filter\_type filter output and stores the value in the filtered variable. That value is available for reuse as much as we want until the endwith tag.{% with filtered = findings|filter\_type([“Network”, “Web”]) %}{% if filtered %}{% set web = filtered|selectattr(“finding\_type”, “==”, “Web”)|list|length %}{% set network = filtered|selectattr(“finding\_type”, “==”, “Network”)|list|length %}

There are {{ network }} *Network* and {{ web }} *Web* findings.

|  |  |
| --- | --- |
| Finding | Severity |
| {%tr for finding in filtered %} | |
| {{ finding.title }} | {% cellbg finding.severity\_color %}{{ finding.severity }} |
| {%tr endfor %} | |

{% endif %}{% endwith %}The filtered, web, and network variables are gone now, so trying to use it here would cause a linting error.

{% for finding in findings %}

{{ finding.title }}

**Severity – {{ finding.severity\_rt }}**

#### Affected Entities

{{p finding.affected\_entities\_rt }}

#### Description

{{p finding.description\_rt }}

#### Impact

{{p finding.impact\_rt }}

#### Mitigation

{{p finding.recommendation\_rt }}

#### Replication Steps

{{p finding.replication\_steps\_rt }}

#### References

{{p finding.references\_rt }}

{% endfor %}