loop关键字说明

在ansible 2.5及以前的版本当中，所有的循环都是使用 `with_X` 风格。但是从2.6版本开始，官方开始推荐使用 `loop` 关键字来代替 `with_X` 风格的关键字。

在playbook中使用循环，直接使用loop关键字即可。

如下示例，启动httpd和postfix服务：

```
tasks:
  - name: postfix and httpd are running
    service:
      name: "{{ item }}"
      state: started
    loop:
      - postfix
      - httpd
```

那么在这个示例当中，其实就是使用loop代替了 `with_list` 循环。

事实上，我们可以使用loop关键字搭配一些过滤器来替换更多的、更复杂的 `with_X` 循环。

with_list

loop可以替代with_list，当处理嵌套列表时，列表不会被拉平

```
- hosts: test
  gather_facts: no
  vars:
```

```
    testlist:
      - a
      - [b,c]
      - d
  tasks:
    - debug:
        msg: "{{ item }}"
      loop: "{{ testlist }}"
```

with_flattened

将所有嵌套都拉平

```
- hosts: test
  gather_facts: no
  vars:
    testlist:
      - a
      - [b,c]
      - d
  tasks:
    - debug:
        msg: "{{ item }}"
      loop: "{{ testlist| flatten }}"
```

with_items

只拉平第一层

```
- hosts: test
  gather_facts: no
  vars:
    testlist:
      - a
      - [b,c]
      - d
  tasks:
    - debug:
        msg: "{{ item }}"
      loop: "{{ testlist| flatten(levels=1) }}"
```

with_indexed_items

通过flatten过滤器（加参数），再配合`loop_control`关键字，可以替代`with_indexed_items`，当处理多层嵌套的列表时，只有列表中的第一层会被拉平

```
- hosts: test
  gather_facts: no
  vars:
    testlist:
      - a
      - [b,c]
      - d
  tasks:
    - debug:
        msg: "{{ index }}: {{ item }}"
      loop: "{{ testlist | flatten(levels=1) }}"
      loop_control:
        index_var: index
```

参数说明：

- `loop_control`： 用于控制循环的行为，比如在循环时获取到元素的索引
- `index_var`： `loop_control`的选项，让我们指定一个变量，`loop_control`会将列表元素的索引值存放到这个指定的变量中

with_together

`zip_longest`过滤器配合list过滤器，可以替代`with_together`

```
- hosts: test
  gather_facts: no
  vars:
    testlist1: [a,b]
    testlist2: [1,2,3]
    testlist3: [A,B,C,D]
  tasks:
    - debug:
        msg: "{{ item.0 }} -- {{ item.1 }} -- {{ item.2 }}"
      #with_together:
      #  - "{{ testlist1 }}"
      #  - "{{ testlist2 }}"
      #  - "{{ testlist3 }}"
    - debug:
        # [a,1,A],[b,2,B],['',3,C],['','',D]
        # [a,1,A],[b,2,B]
        msg: "{{ item.0 }} -- {{ item.1 }} -- {{ item.2 }}"
      loop: "{{ testlist1 | zip_longest(testlist2,testlist3) | list}}"
```

当多个列表使用`with_together`进行对齐合并时，如果多个列表的长度不同，则使用最长的列表进行对齐，由于短列表中的元素数量不够，所以使用空值与长列表中的元素进行对齐，`zip_longest`过滤

器也会像 `with_together` 一样，对列表进行组合，但是还需要借助list过滤器，将组合后的数据列表化。

在使用zip_longest过滤器代替with_together关键字时，默认也是使用空值与长列表中的元素进行对齐，但是也可以指定其他的字符串代替空值，如下示例即使用"NONE"代替空值：

```
- debug:
    msg: "{{ item.0 }} - {{ item.1 }} - {{ item.2 }}"
  loop: {{ testlist1 | zip_longest(testlist2,testlist3,filevalue='NONE') | list }}
```

zip_longest默认使用最长的列表长度进行对齐，当有多个列表的长度不同时，如果希望使用最短的列表对齐，则可以使用zip过滤器：

```
- debug:
    msg: "{{ item.0 }} - {{ item.1 }} - {{ item.2 }}"
  loop: {{ testlist1 | zip(testlist2,testlist3) | list }}
```

with_nested/with_cartesian

可使用product过滤器配合list过滤器以替代 `with_nested` 或者 `with_cartesian` 。product过滤器也是需要将组合后的数据进行列表化，所以需要与list过滤器配合使用：

```
- hosts: test
  gather_facts: no
  vars:
    testlist1: [a,b,c]
    testlist2: [1,2,3,4]
  tasks:
    - debug:
        msg: "{{item.0}} --- { item.1}"
      loop: {{ testlist1 | product(testlist2) | list}}
```

with_sequence

使用range过滤器配合list过滤器可以替代with_sequence：

```
- hosts: test
  gather_facts: no
  tasks:
    - debug:
        msg: "{{ item }}"
      loop: "{{ range(0,6,2) | list }}"
```

上例中表示生成数字，从0开始，到6结束，步长为2。但是需要说明的是，range函数的操作不包含结束范围，也就是说上面的循环只会生成0，2，4三个数字，而不包含6。

另外，with_sequence还有格式化的功能：

```
- debug:
    msg: "{{ item }}"
  with_sequence: start=2 end=6 stride=2 format="number is %0.2f"
```

可使用format配合loop实现：

```
- debug:
    msg: "{{ 'number is %0.2f' |format(item) }}"
  loop: "{{range(2,7,2) |list}}"
```

with_random_choice

使用random函数可以替代with_random_choice，由于random函数是随机取出列表中的一个值，并不涉及循环操作，所以并不使用loop关键字：

```
- hosts: test
  gather_facts: no
  vars:
    testlist: [a,b,c]
  tasks:
    - debug:
        msg: "{{ testlist | random }}"
```

with_dict

可使用loop配合dict2items过滤器实现with_dict功能：

```
- hosts: test
  gather_facts: no
  vars:
    users:
      alice: female
      bob: male
  tasks:
    - debug:
        msg: "{{ item.key }} is {{ item.value}}"
      loop: "{{users | dict2items }}"
```

with_subelements

可使用loop配合subelements过滤器替代with_subelements：

```
- hosts: test
  gather_facts: no
  vars:
    users:
      - name: bob
        gender: male
        hobby:
          - Skateboard
          - VideoGame
      - name: alice
        gender: female
        hobby:
          - Music
  tasks:
    - debug:
        msg: "{{ item.0.name }}'s hobby is {{ item.1}}"
      with_subelements:
        - "{{ users }}"
        - hobby
    - debug:
        msg: "{{ item.0.name }}'s hobby is {{ item.1}}"
      loop: "{{ users | subelements('hobby') }}"
```

在循环语句中注册变量

下面是一个register的变量在循环中使用的例子：

```
# cat register_loop.yml
- name: registered variable usage as a loop list
  hosts: test
  tasks:
      - name: ensure /mnt/bkspool exists
        file:
          path: /mnt/bkspool
          state: directory

      - name: retrieve the list of home directories
        command: ls /home
        register: home_dirs

      - name: Show home_dirs results
        debug:
          var: home_dirs.stdout_lines

      - name: add home dirs to the backup spooler
        file:
```

```
          path: /mnt/bkspool/{{ item }}
          src: /home/{{ item }}
          state: link
          force: yes
        loop: "{{ home_dirs.stdout_lines }}"
```

在循环语句中注册变量：

```
- name: Loop Register test
  gather_facts: no
  hosts: test
  tasks:
    - name: Looping Echo Task
      shell: "echo this is my item: {{ item }}"
      loop:
        - one
        - two
      register: echo_results

    - name: Show echo_results variable
      debug:
        var: echo_results
```

执行语句，可以看到变量的返回结果为一个字典列表：

```
ok: [10.1.61.187] => {
    "echo_results": {
        "changed": true,
        "msg": "All items completed",
        "results": [
            {
                "ansible_loop_var": "item",
                "changed": true,
                "cmd": "echo this is my item: one",
                "delta": "0:00:00.004905",
                "end": "2019-06-10 00:23:51.814151",
                "failed": false,
                "invocation": {
                    "module_args": {
                        "_raw_params": "echo this is my item: one",
                        "_uses_shell": true,
                        "argv": null,
                        "chdir": null,
                        "creates": null,
                        "executable": null,
                        "removes": null,
                        "stdin": null,
                        "stdin_add_newline": true,
                        "strip_empty_ends": true,
                        "warn": true
                    }
                },
                "item": "one",
                "rc": 0,
```

```
                "start": "2019-06-10 00:23:51.809246",
                "stderr": "",
                "stderr_lines": [],
                "stdout": "this is my item: one",
                "stdout_lines": [
                    "this is my item: one"
                ]
            },
            {
                "ansible_loop_var": "item",
                "changed": true,
                "cmd": "echo this is my item: two",
                "delta": "0:00:00.004736",
                "end": "2019-06-10 00:23:52.008981",
                "failed": false,
                "invocation": {
                    "module_args": {
                        "_raw_params": "echo this is my item: two",
                        "_uses_shell": true,
                        "argv": null,
                        "chdir": null,
                        "creates": null,
                        "executable": null,
                        "removes": null,
                        "stdin": null,
                        "stdin_add_newline": true,
                        "strip_empty_ends": true,
                        "warn": true
                    }
                },
                "item": "two",
                "rc": 0,
                "start": "2019-06-10 00:23:52.004245",
                "stderr": "",
                "stderr_lines": [],
                "stdout": "this is my item: two",
                "stdout_lines": [
                    "this is my item: two"
                ]
            }
        ]
    }
}
```