pipe symbol (|),

1. Jinja2中自定义filter并且接受一个参数

{{my\_value|myfilter(filter\_name)}}

def myfilter(value, filtername):

if (filtername is 'twice')

return value\*2

else

.....

2. Html的元素ID要以A-Z或者a-z开头，不要以数字或者字符开头。不然有时会控件会有错误。

3.

Jinja2模板自带一些扩展功能，例如 {% continue %} 以及 {% break %} 标签。 要使用的时候需要先启动。如果在Flask中配置的话，首先需要先在Flask中为jinja2去应用这些扩展才可以在模板中认识这些扩展标签。

例如：

from jinja2.ext import loopcontrols

app.jinja\_env.add\_extension(loopcontrols)

4.

模板的继承(inheritance)以及包含（includes）

继承在模板中使用extends关键字

如果我们render的模板为B,而B在开头extends A模板，这样相当于将B模板加入A模板当中进行渲染。

包含在模板中使用includes关键字

如果我们render的模板为B,而B在当中includes A模板，这样相当于将A模板加入B模板当中进行渲染。默认情况下被包含的模板A将获取B模板的整个context，全部变量.

我们也可以使用 without context来特殊的表明，A模板不会获得context

{% include 'header.html' without context %}

备注：在Jinja2.0中，被包含的模板无法获取在原模板中定义的变量。

例如：

{% for box in boxes %}

{% include "render\_box.html" %} 在Jinja2.0中render\_box.html无法获取box, 但是在Jinja2.1以后可以。

{% endfor %}

模板宏(marco)机制：

我们可以在Jinja2的模板中定义宏，宏的目的在于减少简单逻辑代码的重用。在申明以后，可以像函数一样调用。

例如宏申明;

# 至少接受一个参数，value, type, size为含有默认值参数

{% macro input (name, value='', type='text', size=20) %}

<input type="{{type}} “name="{{name}} “value="{{value|e}} “size="{{size}} “> #根据传入的参数产生具体的html元素

{% endmacro %}

然后在接下来当前的模板中便可以传入参数直接调用

<p> {{input('username')}} </p>

产生<p><input type="text” name="username” value=”” size=20></p>

<p> {{input ('password', type='password')}} </p>

产生<p><input type="password” name="password” value=”” size=20></p>

如果宏在不同的模板中定义(我们可以专门为系统建立一个宏目录)，你需要首先使用 import

{% from 'base/macro/submit.macro' import test %}

然后使用

{{test(参数)}}

5.