**一、准备工作**

1、在Application的参数中添加template\_path 配置，指定模板的路径和静态文件的路径

template\_path='templates',

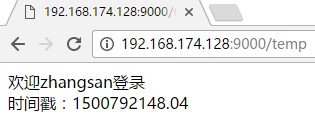
static\_path='static',

2、把模板html文件放入'templates'文件夹下,把静态文件放入'static'文件夹下

3、使用render方法渲染模板

**class** TempHandler(BaseHandler):  
 **def** get(self):  
 name = **'zhangsan'** self.render(**'temp\_index9.html'**,name = name, time = time)

紫色name,time将传到html{{ }}中

<!DOCTYPE **html**>  
<**html lang="en"**><**body**>  
 欢迎{{name}}登录<**br**>  
 时间戳：{{time.time()}}  
</**body**>  
</**html**>

**二、 模板中的符号**

{{   }}   表达式用双大括号包围，内容可以是任何python表达式 ，在双大括号中的单词是占位符

{%   %}   模板控制语句以{%  %}包围 (if,elif,else,end,for)

Tornado里 *<!--{% %}-->* 普通注释对模板符号不管用，

{# {{ }} #} 模板注释

**三、模板控制语句**

控制语句的大部分就像对应的Python语句一样工作

控制语句的返回以最近的{% end  %} 结束

if条件判断

{% if condition %}

     ...

{% elif condition %}

     ...

{% else %}

     ...

{% end %}

for  循环

{% for i in range %}

     ...

{% end %}

**class** TempHandler(BaseHandler):  
 **def** get(self):  
 name = **'zhangsan'** bb = [  
 (**'http://www.baidu.com'**, **u"百度"**),  
 (**'http://www.sohu.com'**, **u"新浪"**),  
 (**'http://www.163.com'**, **u"网易"**)  
 ]  
 self.render(**'temp\_index9.html'**,name = name,bb = bb)

<!DOCTYPE **html**>  
<**html lang="en"**><**body**>  
 欢迎{{name}}登录<**br**>  
 {% for a in bb %}  
 <**a href="{{a[0]}}"**>{{a[1]}} {{a[0]}}</**a**><**br**>  
 {% end %}  
</**body**>  
</**html**>



**四、模板转义**

Tornado默认会自动转义模板中的内容，把标签转换为相应的HTML实体。这样可以防止后端为数据库的网站被恶意脚本攻击。比如，你的网站中有一个评论部分，用户可以在这里添加任何他们想说的文字进行讨论。虽然一些HTML标签在标记和样式冲突时不构成重大威胁（如评论中没有闭<h1>标签），但<script>标签会允许攻击者加载其他的JavaScript文件，打开通向跨站脚本攻击、XSS或漏洞之门。

1、使用raw输出不转义内容

{% raw aaa %}

2、在文件开头使用，这个文件将不进行转义

{% autoescape None %}

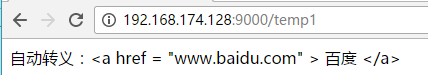
在不进行转义的文件可以使用escape（）进行局部转义

{{ title }}

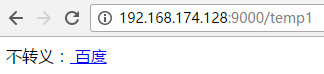
{{ escape(title) }}

3、也可以在Application构造函数中设置autoescape=None，这样设置全局都不转义

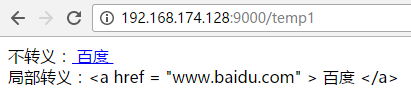
**class** Temp1Handler(BaseHandler):  
 **def** get(self):  
 html = **u'<a href = "www.baidu.com" > 百度 </a>'** self.render(**'temp\_index9.html'**, html = html)

<!DOCTYPE **html**>  
<**html lang="en"**><**body**>  
 自动转义：{{ html }}  
</**body**>  
</**html**>

1、使用raw输出不转义内容

<!DOCTYPE **html**>  
<**html lang="en"**><**body**>  
 不转义：{% raw html %}  
</**body**>  
</**html**>

2、局部转义

<!DOCTYPE **html**>  
{% autoescape None %}  
<**html lang="en"**><**body**>  
 不转义：{{ html }}<**br**>  
 局部转义：{{ escape(html) }}  
</**body**>  
</**html**>

五、**static\_url函数，具体用的时候根据项目来。**

static\_url的函数来生成static目录下文件的URL

<link rel="stylesheet"type="text/css" href="{{ static\_url('css/bootstrap.min.css') }}"/>

大部分这么些：

<link rel="stylesheet"type="text/css" href="/static/css/bootstrap.min.css" />

缺点：但是遇到大项目每次都要算hash值，会很耗时。

<link rel="stylesheet" type="text/css" href="/static/css/bootstrap.min.css

?v=23fdd4e60d5bca15ab7fcbb70446dcec" />

优点：

* 那么为什么使用static\_url而不是在你的模板中硬编码呢？有如下几个原因。其一，static\_url函数创建了一个基于文件内容的hash值，并将其添加到URL末尾（查询字符串的参数v）。这个hash值确保浏览器总是加载一个文件的最新版而不是之前的缓存版本。无论是在你应用的开发阶段，还是在部署到生产环境使用时，都非常有用，因为你的用户不必再为了看到你的静态内容而清除浏览器缓存了。
* 另一个好处是你可以改变你应用URL的结构，而不需要改变模板中的代码。例如，可以通过设置static\_url\_prefix来更改Tornado的默认静态路径前缀/static。如果使用static\_url而不是硬编码的话，代码不需要改变。

**2.2.3 在模板中使用函数**

Tornado在所有模板中默认提供了一些便利的函数。它们包括：

**escape(s)**

替换字符串s中的&、为他们对应的HTML字符。

**url\_escape(s)**

使用urllib.quote\_plus替换字符串s中的字符为URL编码形式。

**json\_encode(val)**

将val编码成JSON格式。（在系统底层，这是一个对json库的dumps函数的调用。查阅相关的文档以获得更多关于该函数接收和返回参数的信息。）

**squeeze(s)**

过滤字符串s，把连续的多个空白字符替换成一个空格。

在Tornado 1.x中，模版不是被自动转义的。在Tornado 2.0中，模板被默认为自动转义（并且可以在Application构造函数中使用autoscaping=None关闭）。在不同版本的迁移时要注意向后兼容。

在模板中使用一个你自己编写的函数也是很简单的：只需要将函数名作为模板的参数传递即可，就像其他变量一样。