Python Web开发项目教材（Flask版）

第3章 模板（课堂笔记）

# 虽然我们可以在Flask程序的视图函数中编写HTML代码，但是在实际开发Web项目时，一个完整的页面往往有上百行甚至上千行HTML代码，如果将HTML代码全部写到视图函数中，这样不仅会使项目的代码变得冗余，而且后期也会难以维护。为了规避这种情况，我们通常会将每个页面的HTML代码保存到一个单独的模板文件中，使展示页面的HTML逻辑代码与Python逻辑代码进行分离，实现表现逻辑和业务逻辑分离的效果。本章将针对模板的相关内容进行介绍。

# 模板与模板引擎Jinja2

Flask默认使用的模板引擎是Jinja2，Jinja2是一个基于python的模板引擎，具有强大的模板功能和灵活的扩展性。它除了允许在模板中使用变量之外，还允许在模板中使用过滤器、选择结构、控制结构、宏等，以多种方式控制模板的输出。

Flask程序的模板是一个HTML文件，模板文件中除了包含固定的HTML代码外，还可以包含描述数据如何插入HTML代码的动态内容，这些动态内容往往会按照模板引擎支持的特殊语法来标记为变量。模板引擎用于识别模板中的特殊语法标记，它会结合请求上下文环境将变量替换为真实的值，生成最终的HTML页面，这个过程称为渲染。

Flask模板引擎Jinja2的工作原理是：在Flask应用中渲染模板时，Flask会将模板和数据结合，生成最终的HTML响应。

1. 、Flask应用程序加载模板引擎并配置相关设置。
2. 、当用户请求一个包含模板的页面时，Flask会找到相应的路由处理函数。
3. 在路由处理函数中，使用render\_template函数指定要渲染的模板文件
4. Flask将模板文件路径传递给Jinja2引擎。
5. Jinja2引擎解析模板文件，查找其中的特殊标记和语法。引擎根据模板文件中的动态内容生成响应的HTML代码，将生成的HTML响应返回给Flask应用程序。由应用程序发送给客户端浏览器。

在FLask程序中使用模板一般分为两个步骤，第一步是创建模板，第二步是使用Jinja2模板引擎渲染模板。

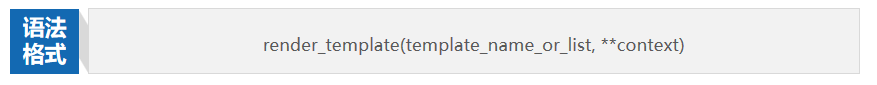
1. **创建模板**

创建模板其实就是创建HTML文件。为了保证Flask程序能够加载到模板文件，我们需要在项目的根目录下新建一个templates文件夹，之后将程序中用到的所有模板文件存放到该文件夹中。

值得一提的是，templates是Flask预先定义好的模板文件夹名称，如果希望使用其他的文件夹名称，则可以在通过代码创建Flask类对象时为template\_folder参数传入其他的文件夹名称。

1. Jinja2模板引擎渲染模板

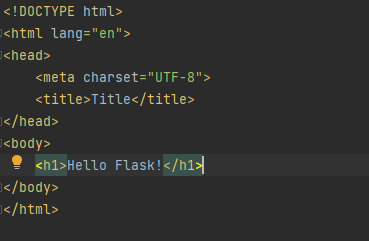
为了能够使用Jinja2模板引擎渲染模板，flask库中提供了render\_template()函数。



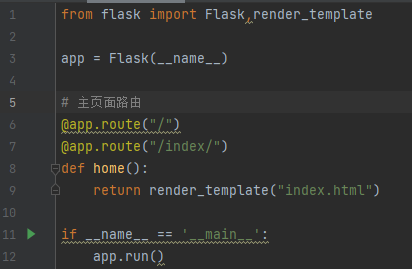
* template\_name\_or\_list：必选参数，表示要加载的模板名称。
* \*\*context：可选参数，表示向模板文件传递的参数，以关键字参数的形式进行传递。注意，关键字参数的名称必须与模板文件中的变量名称保持一致。

通过一个案例分步骤演示如何在Flask程序中使用模板。

通过PyCharm工具创建一个Flask程序Chapter03，在Chapter03项目的根目录下新建templates文件夹，在该文件夹下创建模板文件index.html，并在index.html文件中编写HTML代码。



在Chapter03项目中创建一个app.py文件，并在该文件中先定义一个视图函数， 再在视图函数中渲染模板文件index.html。



# 模板的基础语法

**利用Jinja2这样的模板引擎，我们可以将一部分的程序逻辑放到模板中去。简单地说，我们可以在模板中使用Python语句和表达式来操作数据的输出。但需要注意的是，Jinja2并不支持所有Python语法。而且出于效率和代码组织等方面的考虑，我们应该适度使用模板，仅把和输出控制有关的逻辑操作放到模板中。**

**模板语法主要分为两种：变量和标签。**

### 2.1 模板变量

模板变量是一种特殊的占位符，它用于标识模板中会动态变化的数据，即视图函数传递给模板的变量。当模板被渲染时，模板引擎将模板中的变量替换为由视图函数传递的真实数据。语法如下：

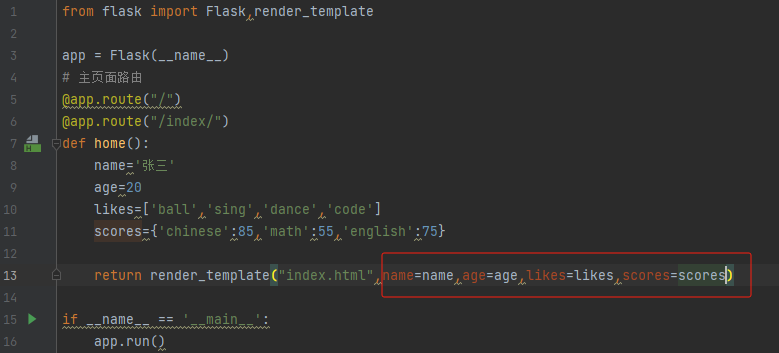


上述格式中，variable表示模板变量的名称，变量名称应与视图函数中传递的关键字参数的名称相同。模板变量的命名规则与Python变量的命名规则相同。注意：变量不存在，默认忽略。

Jinja2能够识别所有类型的变量，例如，列表、字典和对象等。若变量保存的数据是一个列表，则可以通过索引获取列表中的某个元素；若变量保存的数据是一个字典，则可以通过字典的键获取相应的值；

通过示例演示如何在模板中使用模板变量。

在app.py文件的视图函数index()中定义变量，之后将该变量传入render\_template()函数中。



在模板文件index.html编写HTML代码。



请查看执行效果。

### 2.2 模板语句（标签）

在模板文件中，语句使用{%....%}标识，尤其注意的是，在语句结束的地方必须加结束标签

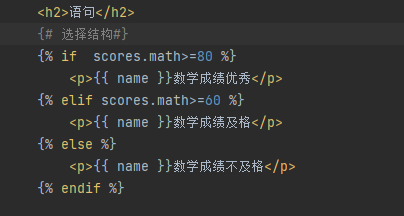
1. 选择结构

Jinja2支持选择结构，选择结构用于判断给定的条件，根据判断的结果执行不同的语句。Jinja2提供了if、elif、else、endif，其中if、elif、else与Python关键字if、elif、else的含义相同，endif用于标识选择结构的末尾。语法如下：



if、elif、else、endif均使用{% %}进行包裹，其中if和endif可以构建最简单的单分支语句，它们与elif、else搭配可以构建更复杂的多分支语句。需要注意的是，选择结构必须以{% endif %}结尾。

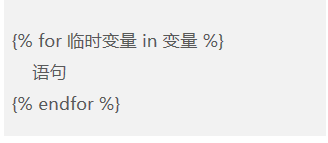
在模板文件index.html添加如下HTML代码。



请查看执行结果

1. 循环结构

Jinja2除了支持选择结构之外，还支持循环结构，其作用是循环遍历变量中的每个元素，以便在模板文件中使用这些元素。Jinja2中循环结构与Python中的for语句用法相似。



for、endfor均使用{% %}进行包裹，其中for标识循环结构的起始位置，endfor标识循环结构的结束位置，且两者都不能省略。

通过一个展示商品列表的案例分步骤演示如何在模板文件中使用循环结构遍历每个商品。

在app.py文件中定义视图函数goods()以及触发该函数的URL规则/goods，然后在视图函数中向模板传递商品列表。

@app.route('/goods')

def goods():

goods\_name = ['洗碗机','电饭锅','电烤箱','电磁灶','微波炉']

return render\_template('loop\_struct.html', goods\_name=goods\_name)

在templates文件夹中新建模板文件loop\_struct.html，并在该文件中使用循环结构遍历模板变量goods\_name取出每个商品名称。

<body>

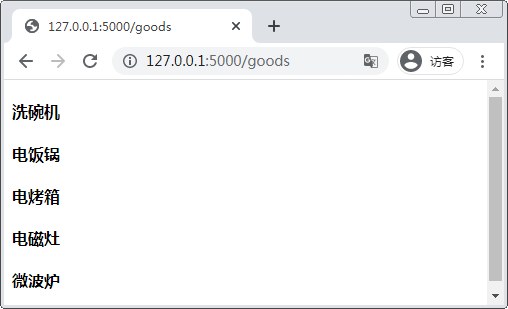
{% for good in goods\_name %}

<h4>{{ good }}</h4>

{% endfor %}

</body>

通过浏览器访问http://127.0.0.1:5000/goods后，页面中显示了循环结构的执行结果。



Jinja2还为循环结构提供了一些特殊的变量，通过访问这些特殊的变量达到控制循环的目的。

|  |  |
| --- | --- |
| **变量** | **说明** |
| loop.index | 当前循环的迭代数（从1开始计数） |
| loop.index0 | 当前循环的迭代数（从0开始计数） |
| loop.revindex | 当前反向循环的迭代数（从1开始计数） |
| loop.revindex0 | 当前反向循环的迭代数（从0开始计数） |
| loop.first | 若当前循环为第一次循环，则返回True |
| loop.last | 若当前循环为最后一次循环，则返回True |
| loop.length | 当前序列包含的元素数量 |
| loop.previtem | 上一个迭代的数据 |
| loop.nextitem | 下一个迭代的数据 |

作业：有学生信息数据如下：

students=[

{'id':'001','name':'谢文婷','sex':'F','birthday':'2005-01-01','phone':'13200000001'},

{'id':'009','name':'陈慧','sex':'F','birthday':'2004-02-04','phone':'13800000005'},

{'id':'003','name':'周忠群','sex':'M','birthday':'2002-06-11','phone':'18900000005'},

{'id':'004','name':'吴鹏','sex':'M','birthday':'2003-10-14','phone':'13500000002'},

{'id':'005','name':'陈颜洁婷','sex':'F','birthday':'2004-08-01','phone':'13001111111'}

]

请将如上数据处理之后在响应给客户端学生信息表如下：



2.3 过滤器

在Jinja2中，过滤器是用于修改或过滤模板变量值的特殊函数，使用过滤器可以获取更精确的数据。



variable表示变量的名称，filter(parameter)表示使用的过滤器，其中parameter表示传递给过滤器的参数，变量名称和过滤器之间使用竖线分隔。如果没有任何参数传给过滤器，则括号可以省略。一个模板变量可以使用多个过滤器，多个过滤器之间使用竖线分隔。

Jinja2提供了许多内置过滤器，常用的内置过滤器如下。

|  |  |
| --- | --- |
| **过滤器** | **说明** |
| abs() | 返回给定参数的绝对值 |
| random() | 返回给定列表中的一个随机元素 |
| safe() | 将变量值标记为安全，保证渲染时不进行转义 |
| tojson() | 将给定参数序列化为JSON字符串 |
| escape() | 用HTML安全序列替换字符串中的字符&、<、>、'和" |
| length() | 返回变量值的长度 |
| sort() | 对变量保存的数据进行排序，该函数内部调用的是Python的sorted()函数 |
| join() | 使用指定符号将字符串中的字符进行拼接，默认符号为空字符串 |
| int() | 将值转换为整数，如果转换不起作用，返回0 |
| float() | 将值转换为浮点数，如果转换不起作用，返回0.0 |
| capitalize() | 将变量值的首字母改为大写字母，其余字母改为小写字母 |
| trim() | 清除变量值前后的空格 |
| upper() | 将变量值转换为大写字母 |

通过一个案例分演示内置过滤器的用法。

在app.py文件中，定义视图函数use\_of\_filters()以及触发该函数的URL规则/filters，在视图函数use\_of\_filters()中向模板传递不同类型的数据。

@app.route('/filters')

def use\_of\_filters():

num = -2.3

li = [2, 1, 5, 6, 7, 4, 4]

string = 'flask'

return render\_template('filters.html', num=num, li=li, string=string)

在templates文件夹中新建filters.html文件，在该文件中对定义的模板变量使用过滤器修改或过滤变量值（部分过滤器演示）。

<body>

{#返回变量num的绝对值#}

<h4>绝对值：{{ num|abs }}</h4>

{#将变量num转换为整型#}

<h4>转换为整型：{{ num|int }}</h4>

{#返回变量li中随机的一个元素#}

<h4>获取随机元素：{{ li|random }}</h4>

{#返回变量li的长度#}

<h4>获取列表长度：{{ li|length }}</h4>

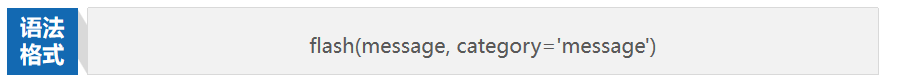
</body>

# 消息闪现

Flask提供了一种良好地向用户反馈信息的方式——消息闪现，消息闪现会在请求完成后记录一条消息，之后在下一次请求向用户提示这条消息。例如，用户在登录页面输入错误的密码后，单击登录按钮才会在页面提示密码错误的消息。

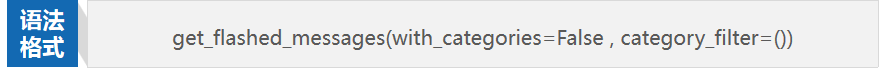
Flask中通过在视图函数中使用flash()函数实现消息闪现的效果，不过flash()函数执行后不会立即在浏览器页面中为用户弹出一条消息，而是需要在模板中通过get\_flashed\_messages()函数获取消息，并将其显示到页面中。

flash()函数通过flask.flash导入使用，用于发送消息。



* message：必选参数，发送闪现的消息。
* category：可选参数，消息的类别。该参数支持4种取值，分别是message、error、info和warning，其中message是默认值，表示任何类型的消息；error表示错误的消息；info表示信息消息；warning表示警告消息。

get\_flashed\_messages()函数是一个全局函数，可在模板的任意位置调用。



* with\_categories：可选参数，表示是否同时返回消息与消息类别，若设置为True，则会以元组形式返回消息和消息类别；若设置为False，则只会返回消息。
* category\_filter：可选参数，表示只返回与消息类别一致的消息。

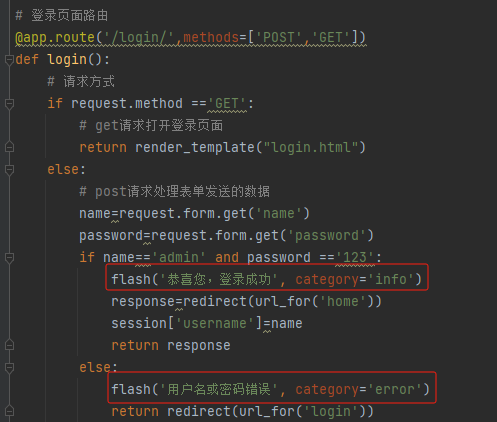
注意：flash()函数会将发送的消息存储到session中，因此我们需要在程序中设置secret\_key。

通过一个用户登录案例为大家分步骤演示如何在Flask程序中实现消息闪现的功能，即如果用户登录成功则跳转到主页面，并在登录页面上提示“恭喜您登录成功”的消息；如果用户登录失败则在登录页面提示“用户名或密码错误”的消息。

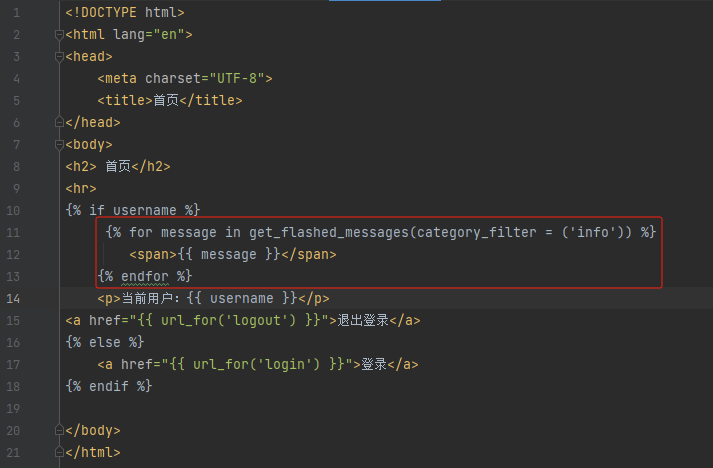
(1)在app.py文件中定义视图函数home\_page()，该函数用于判断用户的登录状态。



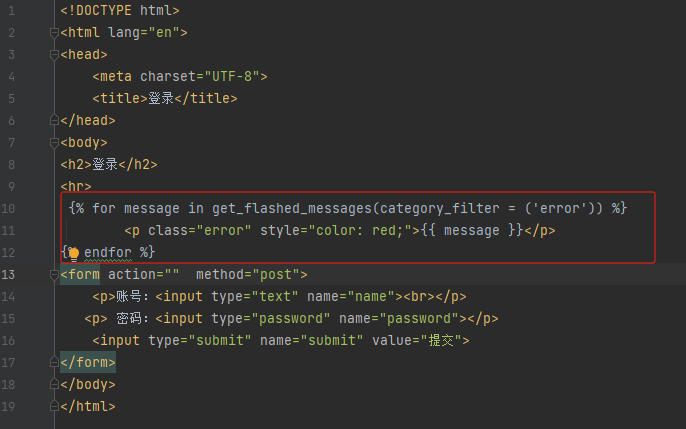
(2)在app.py文件中定义视图函数login()，用于获取用户在login页面输入的用户名和密码，如果用户输入了错误的用户名或密码，则通过消息闪现机制反馈给用户，反之将用户名和密码保存到session中，并将页面重定到home页面。



1. 在templates文件夹中新建模板文件home.html，该模板文件用于展示home页面以及用户登录成功的消息。



1. 在templates文件夹中新建模板文件login.html，该模板文件用于展示login页面以及登录失败的消息。



# 宏的定义与调用

模板中的宏与Python函数类似，它可以传递参数，但没有返回值。在定义宏时，通常会将一部分模板代码写到宏中，然后将代码中动态变化的值替换为模板变量，通过参数传递的方式给变量赋值。

模板中的宏以macro标识开始，以endmacro标识结束。

{% macro 宏的名称(参数列表)%}

宏内部逻辑代码

{% endmacro %}

参数列表中可以有零个、一个或多个参数，多个参数之间使用逗号进行分隔；宏内部可以嵌套使用前文介绍的过滤器、选择结构、循环结构等。

在macro.html文件的<body>标签中，定义一个描述input控件类型的宏。

{% macro inputstyle(name, value='', type='checkbox') %}

<input name="{{ name }}" value="{{ value }}" type="{{ type }}">

{% endmacro %}

宏定义好之后不会立即执行，直到被程序调用时才会执行，程序调用后之后会返回一个包含HTML代码的字符串或模板。



在项目的templates目录下新建文件macro.html，之后在该文件的<body>标签中调用定义好的宏inputstyle。

<h4>您目前对哪些技术感兴趣？</h4>

<p>{{ inputstyle('Python') }} Python</p>

<p>{{ inputstyle('Java') }} Java</p>

<p>{{ inputstyle('big\_data') }} 大数据</p>

<p>{{ inputstyle('JavaScript') }} JavaScript</p>

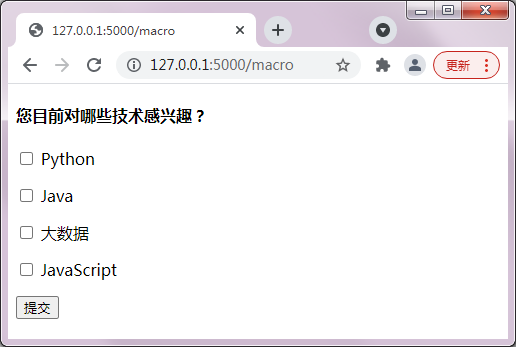
<p>{{ inputstyle('commit', value="提交", type="button") }}</p>

为了能够在页面中看到宏调用后的效果，我们在Chapter03项目的app.py文件中添加渲染模板文件macro.html的代码。

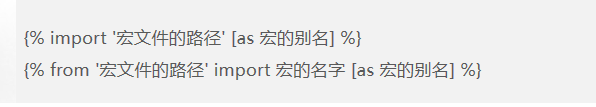
@app.route('/macro')

def input\_style():

return render\_template('macro.html')



如果需要在多个模板文件中使用宏，那么可以将定义宏的代码写入到单独的模板文件中，在调用宏的时候只需要从该模板文件中导入定义的宏即可。宏的导入方式与Python模块的导入方式类似，需要使用import或from... import语句导入定义的宏。



在项目的templates目录下新建文件macro\_called.html，在该文件的<body>标签中导入macro.html中定义的宏inputstyle，并调用宏inputstyle在页面中创建一个复选框。

<body>

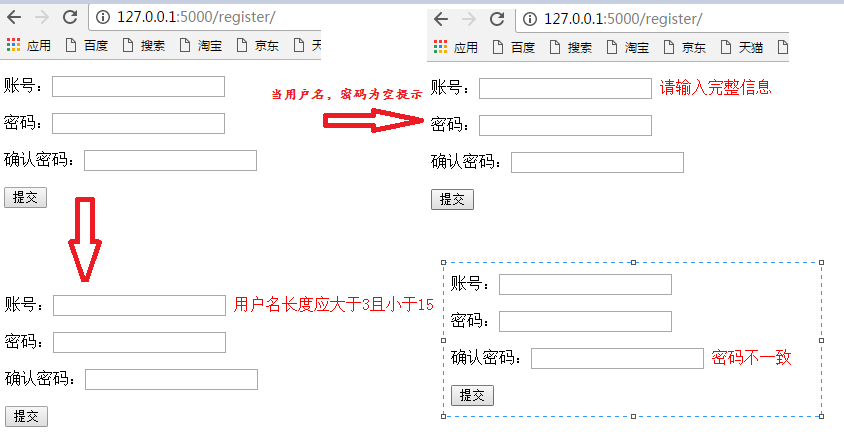
{% from 'macro.html' import inputstyle %}

<p>导入宏</p>

<p>{{ inputstyle('Python') }} Python</p>

</body>

作业：注册页面的实现：展示注册页面以及进行服务器端数据验证（要求用户名，密码不能为空，用户名长度为5-20，密码和确认密码一致）



# 静态文件的加载

在Flask程序中，默认情况下静态文件都存储在与项目文件同级目录的static文件夹中，该文件夹需要由开发人员创建。

为了能够在模板文件中引用静态文件，需要使用url\_for()函数解析静态文件的URL，静态文件的URL规则默认为/static/<path:filename>。url\_for()函数需要接收两个参数，第1个参数表示端点名称，默认值为static；第2个参数filename表示静态文件的名称。



以上代码解析图片文件flask.png的URL规则为/static/test.png。

在模板中若要引用图片文件，可以在定义<img>标签时通过src属性规定显示图像的URL，该URL是调用url\_for()函数解析静态文件的URL。



CSS文件主要用于控制网页的版式、颜色、字体大小和格式等。在模板中若要引用CSS文件，可以在定义<link>标签时通过src属性规定当前模板与CSS文件之间的关系，src属性的值为stylesheet，表示样式表；通过href属性规定被链接文档的URL，该URL是调用url\_for()函数解析静态文件的URL。

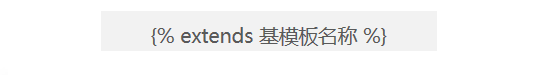


# 模板继承

一个Web网站的多个网页中往往包含一些通用内容和样式，例如，导航栏、标题、页脚等，为了避免在多个模板重复编写通用内容和样式的代码，提高代码的重用率，减少开发人员的工作量，Jinja2提供了模板继承机制。

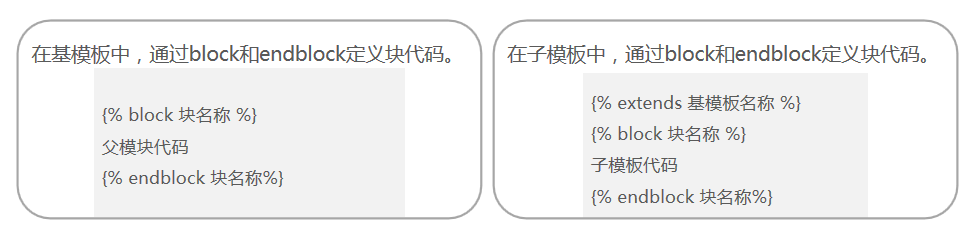
模板继承机制允许开发人员在一个模板中先定义多个页面的通用内容和样式，再以该模板为基础来拓展模板，此时包含通用内容和样式的模板称为基模板，继承基模板的模板称为子模板。

在Jinja2中，模板继承通过block和extends实现，其中block用于标识与继承机制相关的代码块，extends用于指定子模板所继承的基模板。子模板通过继承可以获取基模板中的内容和结构。



{% extends %}必须位于子模板的第一行，当模板引擎解析到{% extends %}时，会将基模板中的内容完整的复制到子模板中。

为了能够让子模板便于覆盖或插入内容到基模板中，我们需要在基模板使用block和endblock定义块代码，同时在子模板中通过定义同名的块。



通过一个案例分步骤演示如何在Flask程序中实现模板继承的功能。

（1）将创建的base.html作为基模板，并在该模板文件中添加创建导航栏和页脚的代码。

<body>

<p>Flask-Logo图片</p>

<img src="{{ url\_for('static', filename='flask.png') }}">

<div class="navigation">

<p>我是导航栏</p>

</div>

<div class="content">

{% block content %}

<p>基模板中的内容</p>

{% endblock %}

</div>

<div class="footer">

<p>我是页脚</p>

</div>

</body>

（2）定义一个继承base.html模板的子模板child.html，子模板中的代码如下所示。

<head>

<meta charset="UTF-8">

<title>子模板</title>

</head>

<body>

{% extends 'base.html' %}

{% block content %}

<p>我是子模板</p>

{% endblock %}

</body>

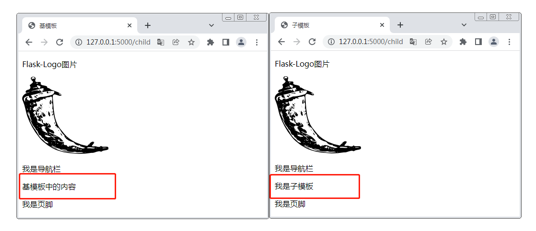
（3）在app.py文件中定义视图函数extends\_template()，之后在该函数中依次渲染模板文件base.html和child.html。

@app.route('/child')

def extends\_template():

return render\_template('base.html')

return render\_template('child.html')



如果子模板实现了基模板中定义的block，那么子模板block中的代码就会覆盖基模板中的代码，如果想要在子模板中仍然呈现基模板中的内容，那么可以使用super()函数实现。

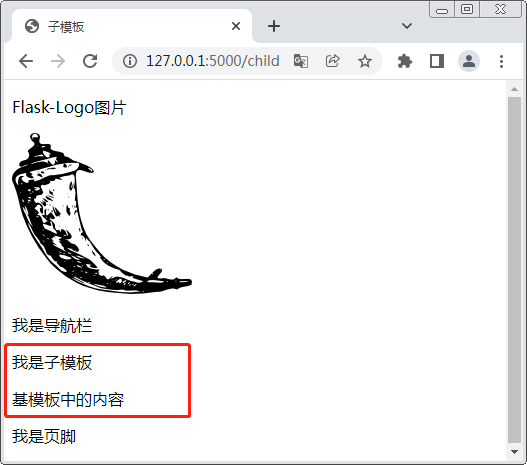
{% extends 'base.html' %}

{% block content %}

<p>我是子模板</p>

{{ super() }}

{% endblock %}



**本章小结**

本章首先介绍了模板与模板引擎Jinja2，然后介绍了模板的基础语法，包括模板变量、过滤器、选择结构和循环结构，接着介绍了宏的定义与调用，最后介绍了消息闪现、静态文件的加载和模板继承。通过学习本章的内容，希望读者能够掌握模板的使用技巧，为后续开发项目奠定扎实的基础。

案例：以读书网为例简单讲解模板的继承和静态文件加载