楼+之Python实战第10期 (/courses/1190)

Jinja2 模板

简介

Jinja2 是一个Python 软件包，实现了 HTML 模板语言。网页的渲染方式一般有两种，一种是后端渲染，一种是前端渲染。后端渲染时，一般都是通过 HTML 模板进行的，模板中可能包含若干逻辑，比如继承自其它基础模板。这些模板逻辑，继承功能都需要模板语言的支持，而 Jinja2 正是这样的一种语言。有了 Jinja2 以后，HTML 模板的编写将变得非常简单。在本节实验中，我们将学习 Jinja2 的方方面面，比如模板变量，循环功能，过滤器等等。

知识点

- Jinja 语法
- Jinja 基础
- Jinja 模板
- Jinja 过滤器

Jinja 语法

模板一般和 Web 框架配合使用，Flask 的默认模板功能就是通过 Jinja2 实现的。所以通过 Flask 学习 Jinja 再好不过。

通过下面的命令创建 Flask app:

```
$ cd /home/shiyanlou/Code
$ mkdir templates
$ echo "<h1>hello world</h1>" > templates/index.html
$ touch app.py
```

以上命令，创建了 `templates` 目录，Flask 默认已经整合了 Jinja, 会自动从 `templates` 目录加载相应的模板。接着在 `app.py` 中输入代码：

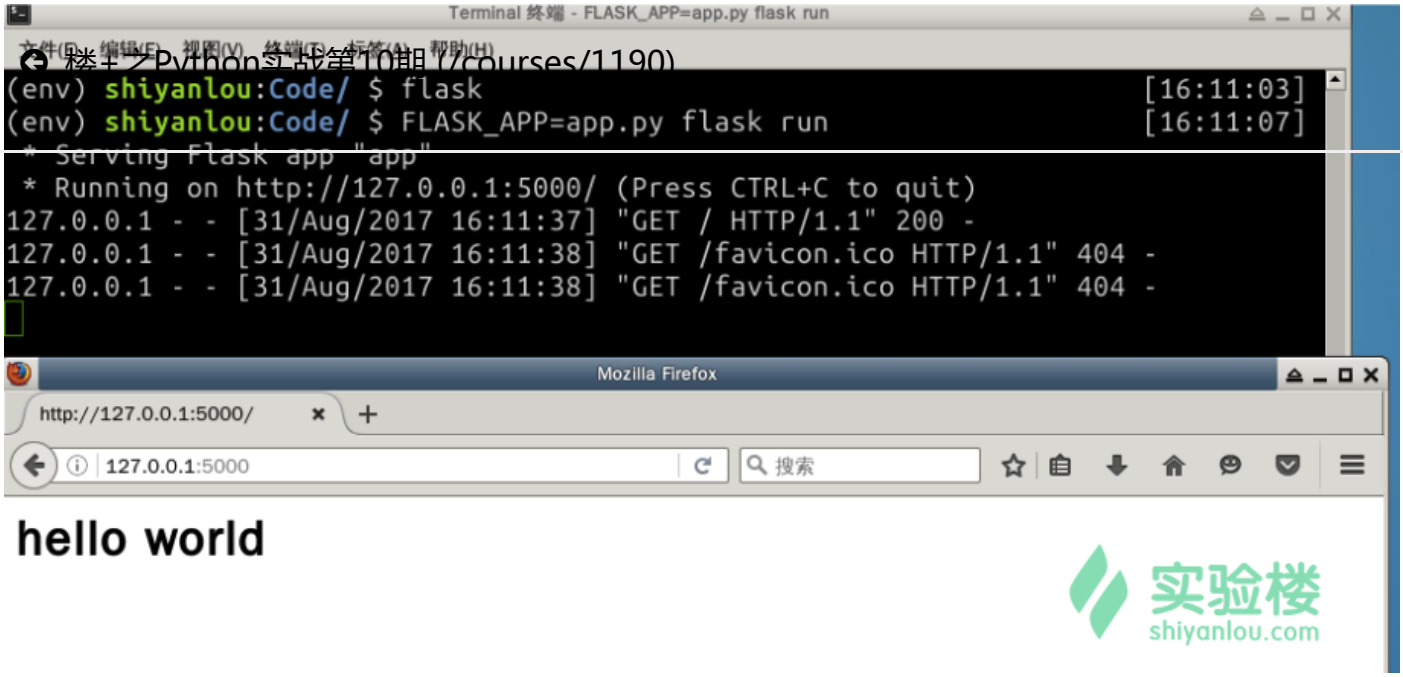
楼+之Python实战第10期 (/courses/1190)

```
#!/usr/bin/env python
# -*- coding:utf-8 -*-

from flask import Flask, render_template


app = Flask(__name__)
app.config['TEMPLATES_AUTO_RELOAD'] = True

@app.route('/')
def index():

    teacher = {
        'name': 'Aiden',
        'email': 'luojin@simplecloud.cn'
    }

    course = {
        'name': 'Python Basic',
        'teacher': teacher,
        'user_count': 5348,
        'price': 199.0,
        'lab': None,
        'is_private': False,
        'is_member_course': True,
        'tags': ['python', 'big data', 'Linux']
    }
    return render_template('index.html', course=course)
```

上面的代码中，创建了一个 Flask 应用，需要注意的是我们设置了 `app.config['TEMPLATES_AUTO_RELOAD'] = True`，这使得每当模板发生改变时，会自动重新渲染模板。接着创建了 `index` 视图函数，其中定义了 `course` 字典，代表一门课程。该视图函数会渲染 `index.html` 模板，该模板位于 `/home/shiyanlou/Code/templates/index.html` ，后续关于 Jinja 的演示代码都会写入 `index.html` 文件中，然后通过浏览器查看效果。通过以下命令启动 Flask 应用:

```
$ cd /home/shiyanlou/Code
$ FLASK_DEBUG=1 FLASK_APP=app.py flask run
* Serving Flask app "app"
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

启动应用时，通过环境变量 `FLASK_DEBUG=1` 设置 Flask 启动在 debug 模式。启动成功后，可以通过浏览器访问 `http://127.0.0.1:5000/` ，效果如下：



楼主之Python实战第10期 (/courses/1190)

Flask 与 Jinja 结合开发视频：

0:00 / 3:56

Jinja 基础

Jinja2 的语法中，使用一些特殊字符包含需要解析执行的代码，没有被这些特殊字符包含的代码则不进行解析处理。主要包括以下几种特殊字符（`...` 字符代表省略的需要执行的代码）：

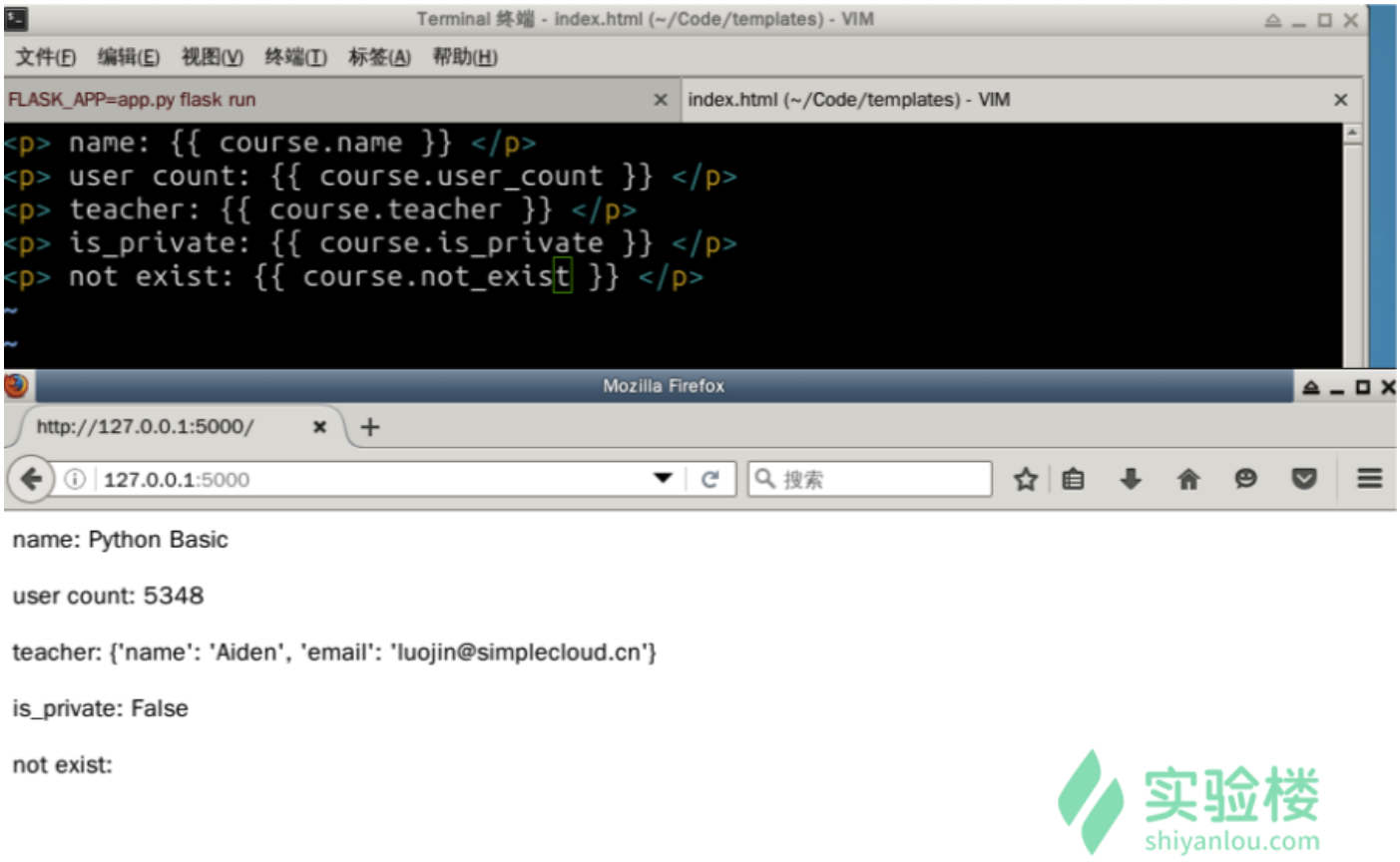
- `{% ... %}` 包含的代码是可以被执行的语句，比如循环语句，继承语法；
- `{{ ... }}` 包含的是 Python 对象，用于解析出这些对象的值，经常用于打印内容；
- `{# ... #}` 用于添加注释，这些注释不会被处理，但是也不会输出到 HTML 源码中；

变量

在 Jinja2 中，变量可以通过 `{{ variable }}` 的形式显示，可以通过 `.` 访问变量的属性，如果传递给模板的变量是一个字典，也可以通过 `.` 访问字典的字段。在前文中，我们创建的 `app.py` 文件中，已经将一个 `course` 对象传递给了 `index.html` 模板，现在将 `index.html` 模板修改成以下内容，重新刷新浏览器就可以看到 `course` 的各种属性了：

```
<p> name: {{ course.name }}</p>
<p> user count: {{ course.user_count }}</p>
<p> teacher: {{course.teacher }} </p>
<p> is_private: {{ course.is_private }} </p>
<p> not exist: {{ course.not_exist }} </p>
```

效果图：



还可以发现，如果访问一个不存在的属性，则会简单的返回空值，不会发生异常，这与在 Python 代码中访问不存在的属性是不一样的。

Jinja2 也支持赋值操作，有的时候执行方法可能消耗很多资源，这个时候可以将执行结果通过 `set` 关键字赋值给 Jinja2 变量，在后续的所有访问中，都通过该变量访问，如下代码：

```
{% set result = heavy_operation() %}

<p> {{ result }}</p>
```

练习题：渲染出变量值

> 注意：练习题的目录和学习的目录是分开的，本节实验中的练习题应该在 `/home/shiyanlou/flasktest/` 目录下操作，而操作演示的代码在 `/home/shiyanlou/Code` 目录下执行

首先使用 Ctrl + C 停止前面运行的 flask run 命令，然后进入 flasktest 目录下：

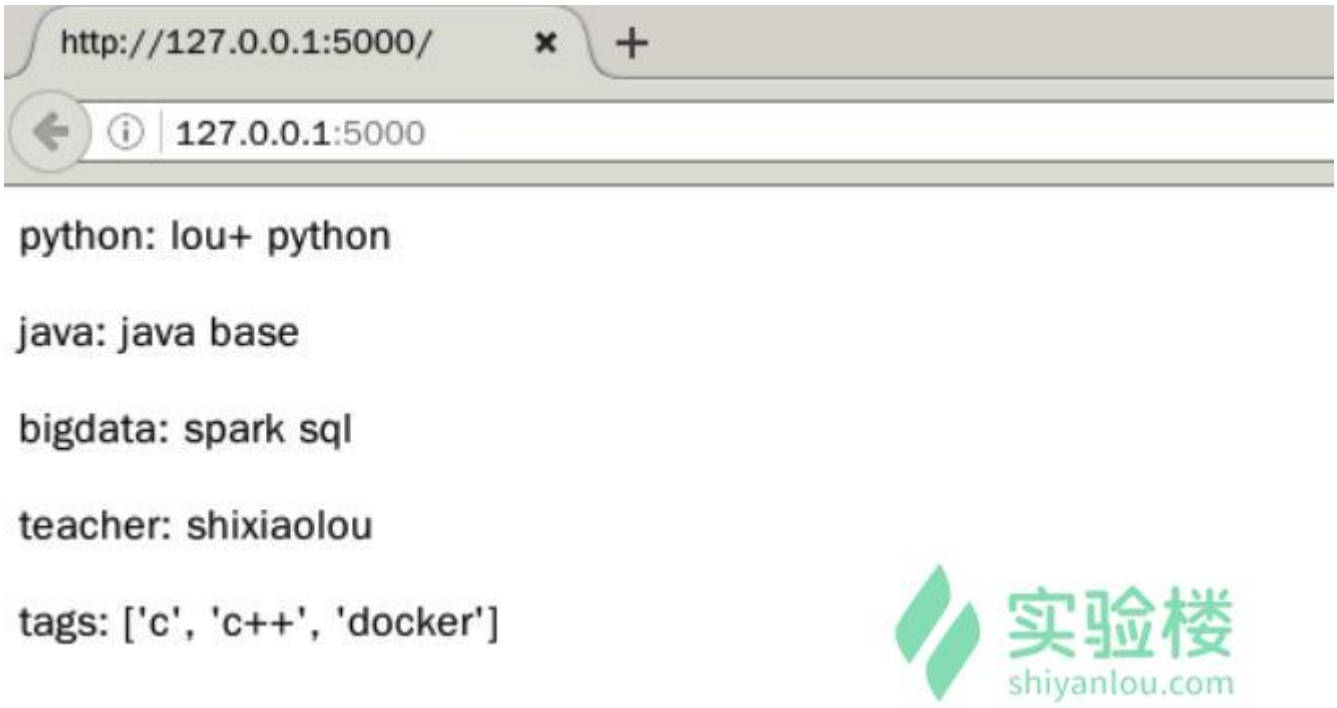
楼+之Python实战第10期 (/courses/1190)

```
cd /home/shiyanlou/flasktest
```

修改 /home/shiyanlou/flasktest/app.py 中 index 函数如下：

```
@app.route('/')
def index():
    course = {
            'python': 'lou+ python',
            'java': 'java base',
            'bigdata': 'spark sql',
            'teacher': 'shixiaolou',
            'is_unique': False,
            'has_tag': True,
            'tags': ['c', 'c++', 'docker']
            }
    return render_template('index.html',course=course)
```

新建 /home/shiyanlou/flasktest/templates/index.html 页面，并向其中写入代码，要求实现当应用运行后，我们访问 http://localhost:5000/ 的时候，可以看到如下的内容显示在页面上。



完成代码后，需要在终端按照之前的所学配置 FLASK_APP 环境变量并使用 flask run 启动应用。

注意本节实验后续的几个题目是连续的，所以 /home/shiyanlou/flasktest 目录创建后不要删除。

应用启动后，点击 下一步 ，系统将自动检测完成结果。

逻辑比较

Jinja 中的逻辑比较可以通过 `if` 语句，如下代码：

楼+之Python实战第10期 (/courses/1190)

```
{% if course.is_private %}
    <p> course {{course.name}} is private </p>
{% elif course.is_member_course %}
    <p> course {{course.name}} is member course </p>
{% else %}
    <p> course {{course.name}} is normal course </p>
{% endif %}
```

可以看到语法和 Python 中的 if 判断差不多，但是需要包裹在 `{% %}` 符号中，同时结尾需要有 `endif` 语句，在 `index.html` 中输入代码后，效果如下图：

循环

在 Jinja 中循环主要通过 `for` 语句完成，语法如下：

```
{% for tag in course.tags %}
    <span> {{ tag }} </span>
{% endfor %}
```

练习题：判断并循环列出 tags

首先使用 `Ctrl + C` 停止前面运行的 flask run 命令，然后进入 flasktest 目录下：

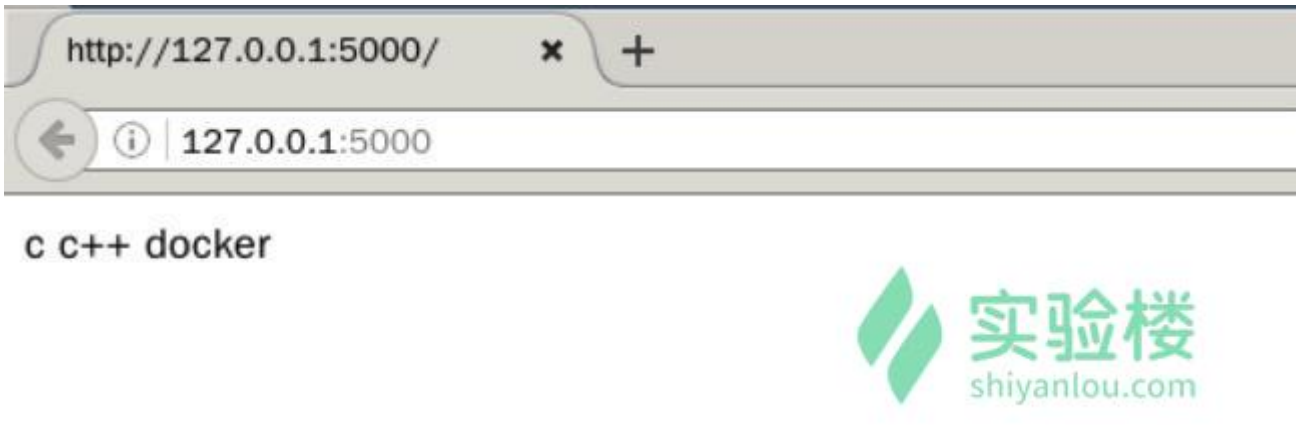
楼+之Python实战第10期 (/courses/1190)

```
cd /home/shiyanlou/flasktest
```

修改 `/home/shiyanlou/flasktest/templates/index.html` 页面，向其中写入代码，要求实现如下需求：

- 判断 course 是否为 unique，如果是则页面显示： `course is unique` 。
- 判断 course 是否有 tag，如果有则循环遍历每一个标签，在页面中依次显示标签的值。
- 如果以上两者都不是，则页面显示： `course is not unique and it don\'t has tag` 。

完成代码后，需要在终端按照之前的所学配置 FLASK_APP 环境变量并使用 flask run 启动应用。我们访问 `http://localhost:5000/` 的时候，可以看到如下的内容显示在页面上。



注意本节实验后续的几个题目是连续的，所以 `/home/shiyanlou/flasktest` 目录创建后不要删除。

应用启动后，点击 下一步 ，系统将自动检测完成结果。

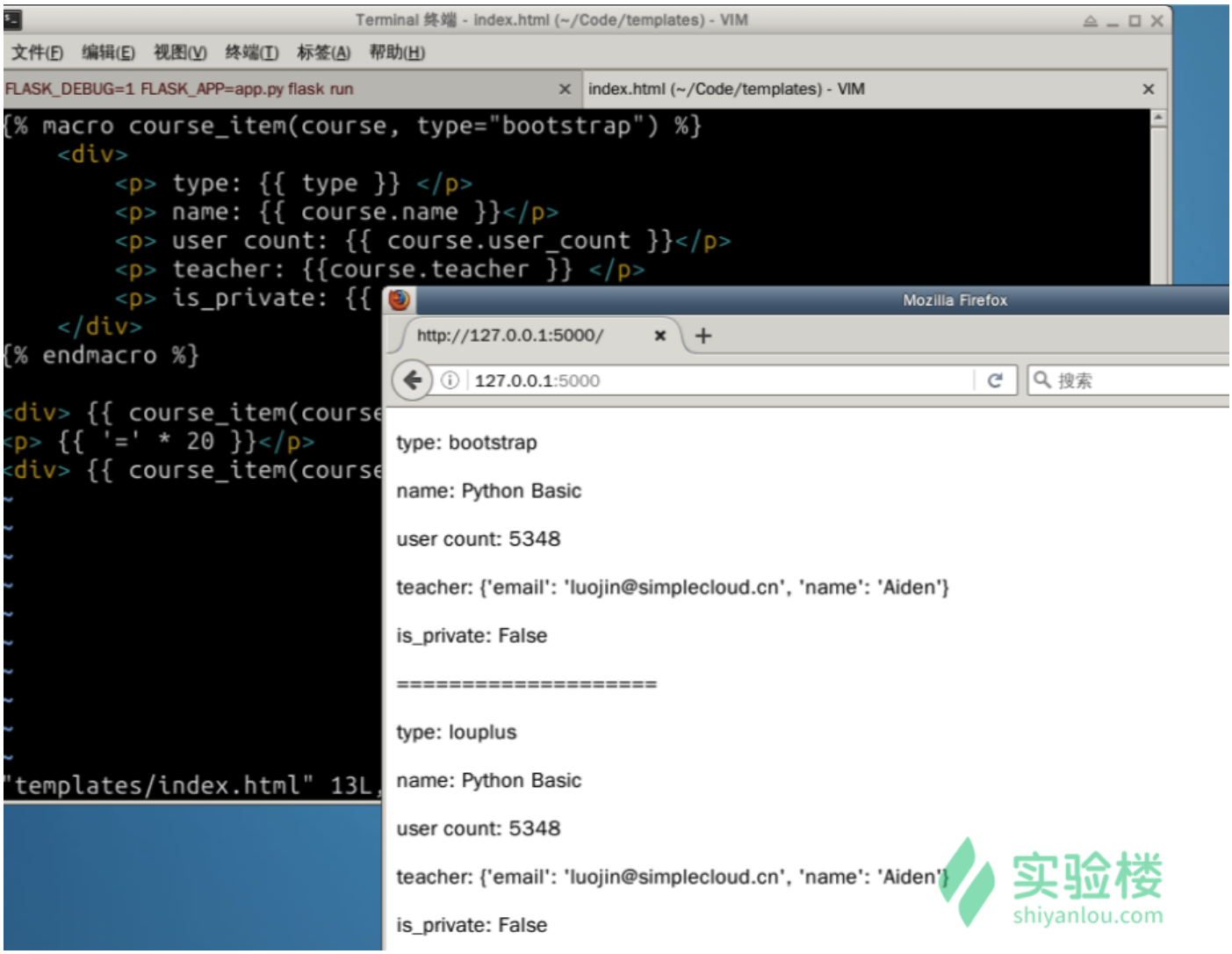
宏

在 Python 中可以定义各种函数，同样的在 Jinja2 中，可以定义宏，相当于 Python 中的函数。可以将常用的 HTML 代码写成一个宏，这样在任何地方调用宏就可以生成同样的 HTML 代码，提高了代码复用度。宏通过 `macro` 关键字进行定义。比如可以将渲染一门课程信息的代码写成一个宏:

楼+之Python实战第10期 (/courses/1190)

```
{% macro course_item(course, type="bootstrap") %}
    <div>
        <p> type: {{ type }} </p>
        <p> name: {{ course.name }}</p>
        <p> user count: {{ course.user_count }}</p>
        <p> teacher: {{course.teacher }} </p>
        <p> is_private: {{ course.is_private }} </p>
    </div>
{% endmacro %}

<div> {{ course_item(course) }} </div>
<p>{{ '=' * 20 }}</p>
<div> {{ course_item(course, type="louplus") }} </div>
```

上面的代码中，定义了 `course_item` 宏，该宏有两个参数，第一个是课程，第二个是类型，且第二个参数具有默认值，和 Python 函数非常像，接着通过 `{{ course_item(course) }}` 方式调用了两次宏。上面代码写入 `index.html`，刷新页面效果如下：



Jinja 宏使用示例视频：

楼+之Python实战第10期 (/courses/1190)

0:00 / 5:41

模块

上文中定义的宏有可能需要被其他模板引用，好在 Jinja2 也支持模块功能。首先我们在 `/home/shiyanlou/Code/templates` 目录中创建 `macro.html` 文件，然后将上文中定义 `course_item` 宏的代码写入该文件。然后就可以在 `index.html` 中通过 `import` 关键字导入宏了，如下代码：

```
{% from 'macro.html' import course_item %}

<div> {{ course_item(course) }} </div>
```

可以发现模块的导入方式和 Python 的也比较类似。

练习题：把代码封装到宏中并通过模块导入的方式使用

首先使用 `Ctrl + C` 停止前面运行的 flask run 命令，然后进入 flasktest 目录下：

```
cd /home/shiyanlou/flasktest
```

在 `/home/shiyanlou/flasktest/templates` 目录下新建 `macro.html` 页面，在其中定义一个名为 `course_tag` 的宏，将上一个练习题中的判断和循环 tag 的代码写入宏中。

修改 `/home/shiyanlou/flasktest/templates/index.html` 页面代码，调用 `macro.html` 页面中的宏，实现和上一个练习题相同的效果。

完成代码后，需要在终端按照之前的所学配置 FLASK_APP 环境变量并使用 flask run 启动应用。

注意本节实验后续的几个题目是连续的，所以 `/home/shiyanlou/flasktest` 目录创建后不要删除。

应用启动后，点击 下一步 ，系统将自动检测完成结果。

模板继承

Jinja2 同样支持模板间的继承。网页中，很多组件是共用的，比如网页的标题栏和尾部，通过继承功能可以很方便的共用组件。继承功能通过 `extends`，`block` 等关键字实现。首先在 `/home/shiyanlou/Code/templates` 目录中创建 `base.html` 模板，然后写入以下代码：

楼v之Python实战第10期 (/courses/1190)

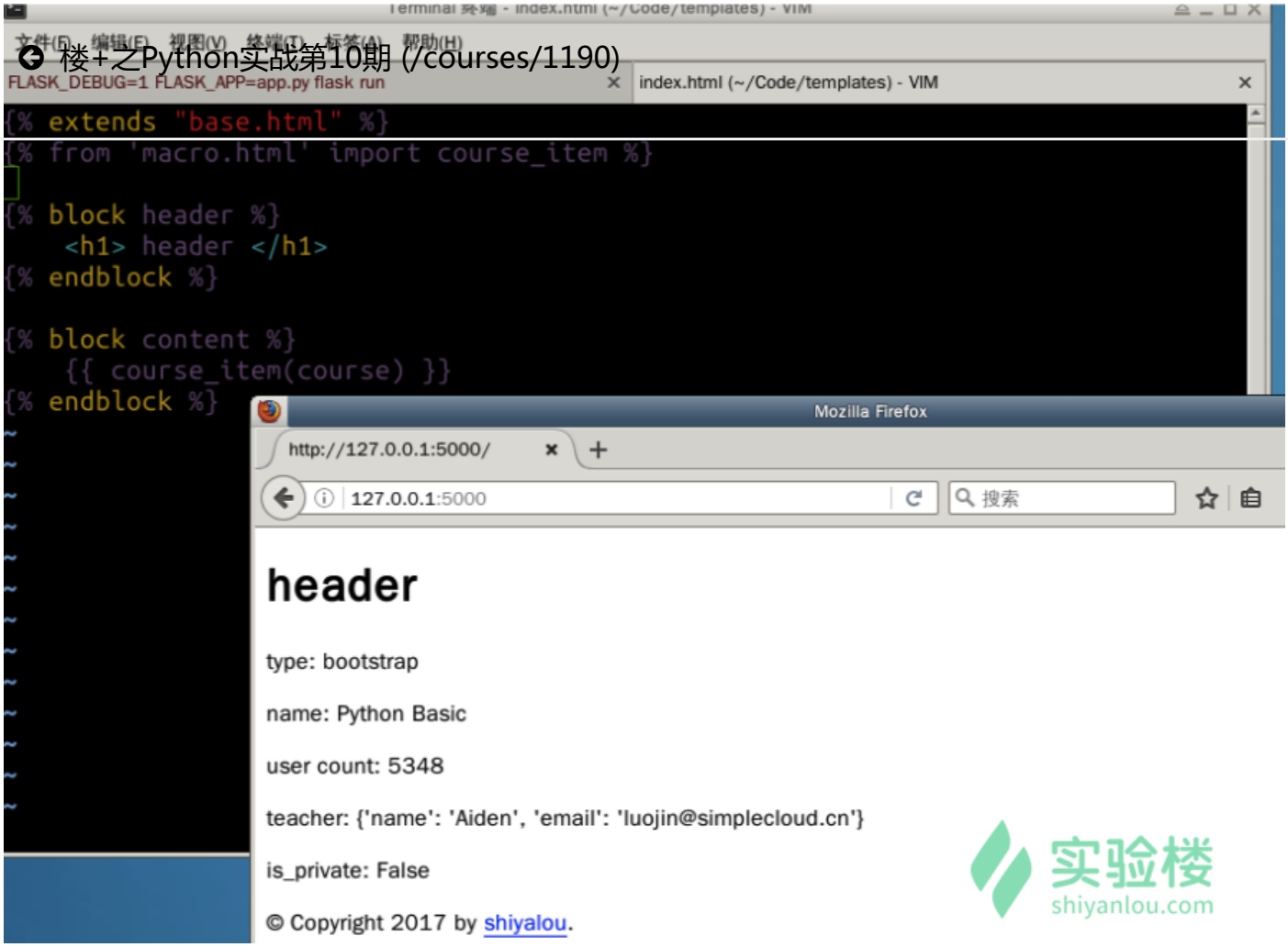
```
    <div>
        {% block header %}
            <p> this is header </p>
        {% endblock %}
    </div>
    <div>{% block content %}{% endblock %}</div>
    <div id="footer">
        {% block footer %}
        &copy; Copyright 2017 by <a href="http://www.shiyanlou.com/">shiyalou</a>.
        {% endblock %}
    </div>
</body>
```

上面的代码中，通过 `block` 定义了 `header`，`content`，`footer` 三个块，可以被其他模板重写，如果未被其他模板重写则显示默认内容。在 `index.html` 中输入以下代码：

```
{% extends "base.html" %}
{% from 'macro.html' import course_item %}

{% block header %}
    <h1> header </h1>
{% endblock %}

{% block content %}
    {{ course_item(course) }}
{% endblock %}
```

上面的代码中，首先通过 `extends` 关键字告诉 Jinja2 模板从 `base.html` 继承而来，接着使用 `import` 关键字从上一节实验中定义的 `macro.html` 导入了宏 `course_item`。接着使用 `block` 关键字覆盖了 `base.html` 模板中定义的 `header`，`content` 块，而 `footer` 块将显示默认内容。刷新浏览器，效果如下图：

楼+之Python实战第10期 (/courses/1190)



Jinja 模块使用视频：

0:00 / 10:03

练习题：继承模板并改写

首先使用 `Ctrl + C` 停止前面运行的 flask run 命令，然后进入 flasktest 目录下：

```
cd /home/shiyanlou/flasktest
```

在 `/home/shiyanlou/flasktest/templates` 目录下新建 `base.html` 页面，向其中写入如下代码：

楼v之Python实战第10期 (/courses/1190)

```
    <div>
        {% block header %}
        <p> header </p>
        {% endblock %}
    </div>
    <div>
        {% block content %}
        <p> content </p>
        {% endblock %}
    </div>
    <p> hello world </p>
    <div>
        {% block footer %}
        <p> footer </p>
        {% endblock %}
    </div>
</body>
```

修改 `/home/shiyanlou/flasktest/templates/index.html` 页面代码，满足如下需求：

- 继承自 base.html 页面，从 macro.html 页面导入 course_tag
- 改写 header 模块，使标题以 h1 显示 shiyanlou
- 改写 content 模块，调用 course_tag 模块
- 改写 footer 模块，使其显示 shiyanlou 且点击后能够跳转到实验楼官网

完成代码后，需要在终端按照之前的所学配置 FLASK_APP 环境变量并使用 flask run 启动应用。最后显示的效果图如下：

shiyanlou

c c++ docker

hello world

shiyanlou

实验楼
shiyanlou.com

注意本节实验后续的几个题目是连续的，所以 `/home/shiyanlou/flasktest` 目录创建后不要删除。

应用启动后，点击 下一步 ，系统将自动检测完成结果。

楼+之Python实战第10期 (/courses/1190)

过滤器

Jinja2 中还支持过滤器，过滤器通过 | 方式执行，比如 `{{ var | abs }}`，通过 `abs` 过滤器对 `var` 求绝对值。 Jinja2 有很多内置的过滤器，比如:

- `abs` 求绝对值；
- `capitalize` 将字符串首字母变成大写，其他转换为小写；
- `first` 获取列表的第一个元素；
- `int` 转换为整数；
- `length` 求列表的长度；

内置的过滤器有很多就不一一列出了。Jinja2 也支持自定义过滤器，通过 Flask 添加过滤器非常简单，修改 `app.py` 为如下代码：

楼+之Python实战第10期 (/courses/1190)

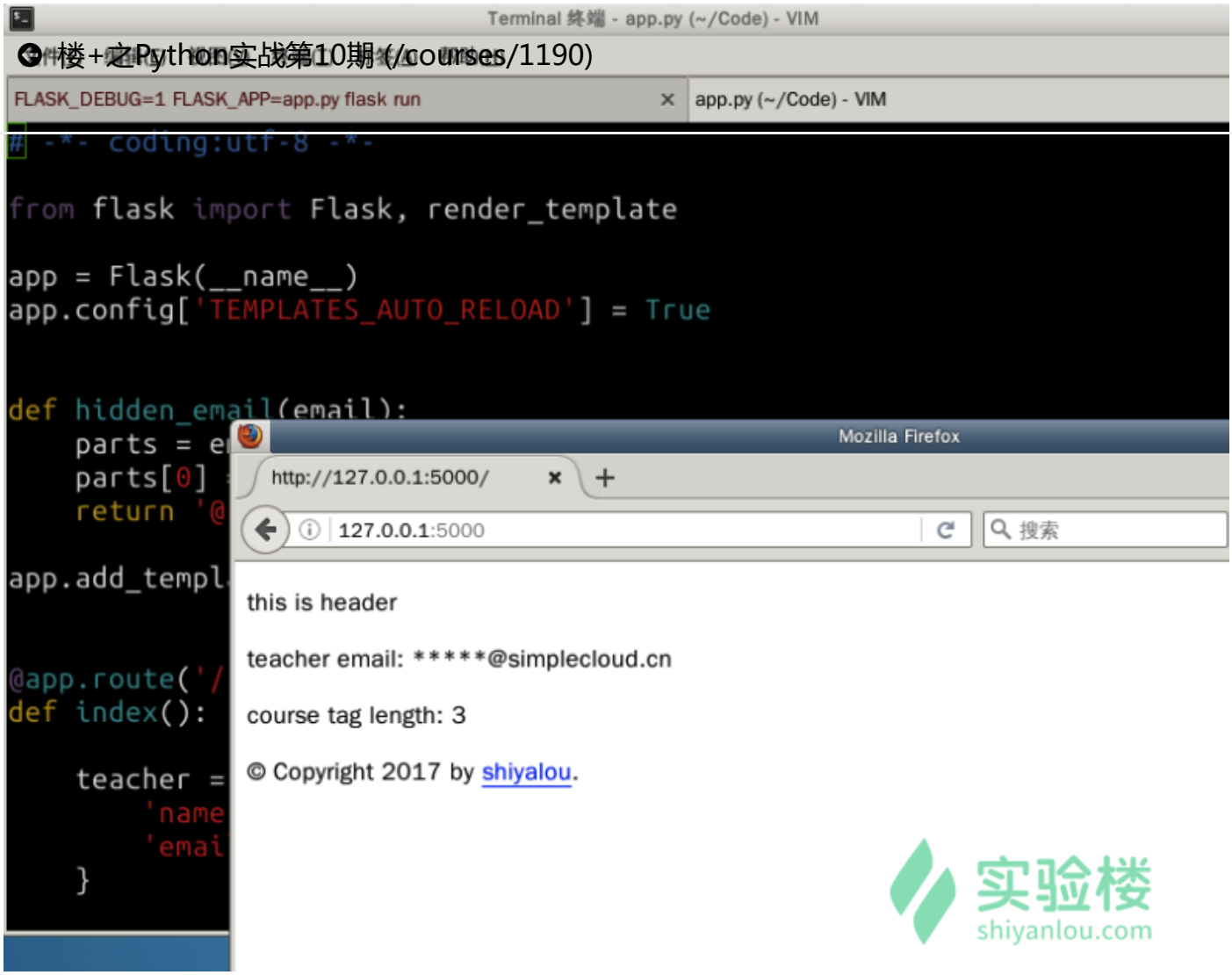
```
# coding: utf8
from flask import Flask, render_template


app = Flask(__name__)
app.config['TEMPLATES_AUTO_RELOAD'] = True

def hidden_email(email):
    parts = email.split('@')
    parts[0] = '*****'
    return '@'.join(parts)

app.add_template_filter(hidden_email)

@app.route('/')
def index():

    teacher = {
        'name': 'Aiden',
        'email': 'luojin@simplecloud.cn'
    }

    course = {
        'name': 'Python Basic',
        'teacher': teacher,
        'user_count': 5348,
        'price': 199.0,
        'lab': None,
        'is_private': False,
        'is_member_course': True,
        'tags': ['python', 'big data', 'Linux']
    }
    return render_template('index.html', course=course)
```

以上代码，通过 `app.add_template_filter` 方法注册了 `hidden_email` 过滤器，该过滤器隐藏邮箱前缀。接着在 `index.html` 文件中输入下面的代码：

```
{% extends "base.html" %}
{% from 'macro.html' import course_item %}

{% block content %}
    <p> teacher email: {{ course.teacher.email | hidden_email }}</p>
    <p> course tag length: {{ course.tags | length }} </p>
{% endblock %}
```

刷新浏览器，效果如下：

楼+之Python实战第10期(/courses/1190)



Jinja 过滤器使用视频：

0:00 / 5:53

url_for

还记得我们在 Flask 入门实验中学习的 `url_for` 来构建 URL 地址的方法吗？同样也可以在 Jinja 中使用，使用方法与 Flask 中相同，但需要在前后增加两个大括号才能在 Jinja 中解析成正确的 URL 地址：

```
{{ url_for('user_index', username='testuser') }}
```

此外，还有一个常用的 static 目录的用法，开发一个 Web 应用的时候需要将一些图片、js、css 等文件放到一个统一的 static 目录，获取这些文件的地址的方式可以用下面的方式：

```
{{ url_for('static', filename='css/style1.css') }}
```

楼+之Python实战第10期 (/courses/1190)

Jinja2 中默认的 static 目录是和 templates 目录同一层次的目录。只要将图片、js、css 等文件放到这个 static 目录下就可以使用这种方法获得文件的 URL 地址了。

练习题：增加一个跳转链接

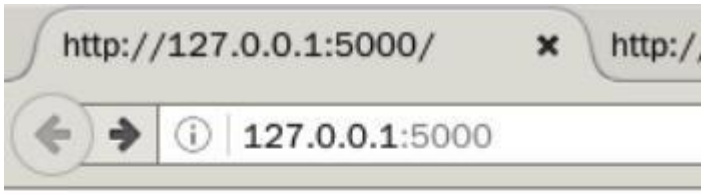
首先使用 `Ctrl + C` 停止前面运行的 flask run 命令，然后进入 flasktest 目录下：

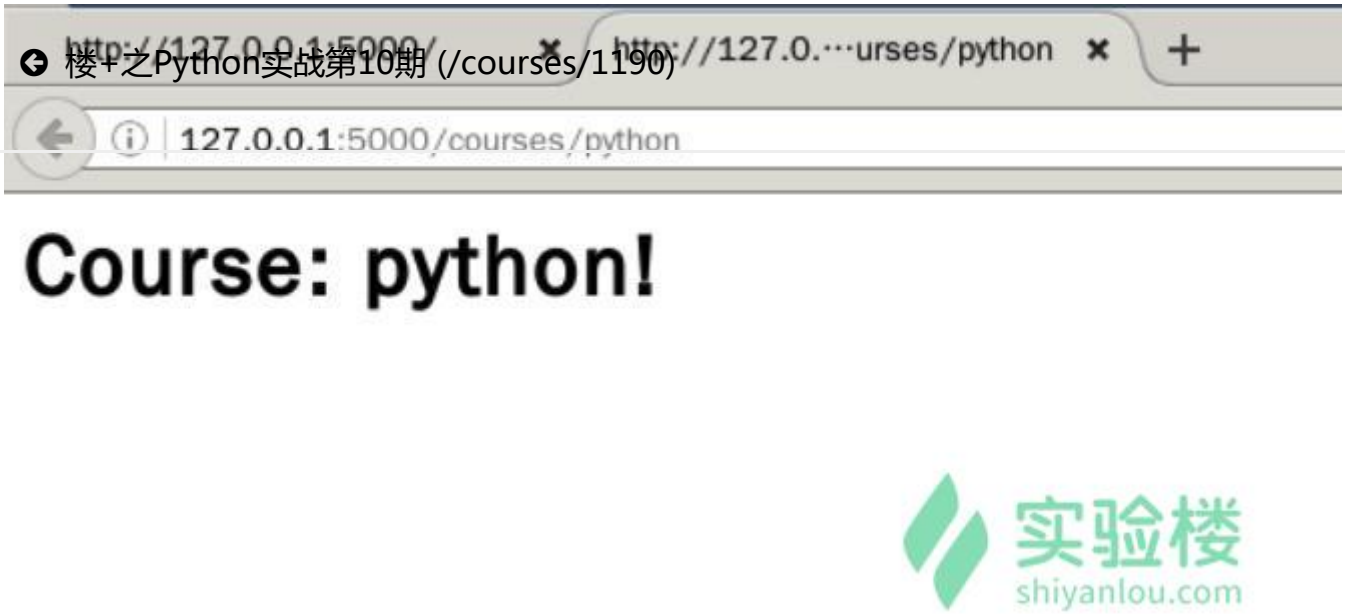
```
cd /home/shiyanlou/flasktest
```

修改 `/home/shiyanlou/flasktest/templates/index.html` 页面代码，满足如下需求：

- 改写 footer 模块，使页面上显示 courses 链接，点击后就可以跳转到 `http://127.0.0.1:5000/courses/python` ，并且跳转后的页面显示 `Course:python!`

完成代码后，需要在终端按照之前的所学配置 FLASK_APP 环境变量并使用 flask run 启动应用。最后显示的效果图如下：





注意本节实验后续的几个题目是连续的，所以 `/home/shiyanlou/flasktest` 目录创建后不要删除。

应用启动后，点击 下一步 ，系统将自动检测完成结果。

总结

本节实验讲解了 Jinja2 的基础知识，覆盖了 Jinja2 的大部分知识点，包括：

- 变量；
- 逻辑比较；
- 循环；
- 宏；
- 模板；
- 模板继承；
- 过滤器；

在实际的项目中，以上功能都可能会被使用，所以请努力掌握相关知识点。楼+ 课程后续会有大量的 Web 开发实战，这个过程中需要写很多 Jinja 模板代码，相对于 CSS/HTML/JavaScript 部分的内容，Jinja 的语法要简单很多。

拓展阅读

- Jinja2 官方文档（中文）(http://docs.jinkan.org/docs/jinja2/)
- Jinja2 官方文档（英文）(http://jinja.pocoo.org/docs/2.10/)

上一节：HTML 和 CSS (/courses/1190/labs/8533/document)

楼+之Python实战第3期(/courses/1190)

挑战：一个简单的资讯网站 (/courses/1190/labs/8535/document)