# Flask 介绍

## 一、实验介绍

### 1.1 实验内容

简单介绍Flask框架的使用

### 1.2 实验知识点

- 微框架、WSGI、模板引擎概念
- 使用 Flask 做 web 应用
- 模板的使用
- 根据 URL 返回特定网页

### 1.3 实验环境

- python3.5
- Xfce终端
- Vim

### 1.4 适合人群

本课程属于初级级别课程，不仅适用于那些有其它语言基础的同学，对没有编程经验的同学也非常友好

## 二、实验步骤

# 2.1 什么是 Flask?

Flask 是一个 web 框架。也就是说 Flask 为你提供工具，库和技术来允许你构建一个 web 应用程序。这个 web 应用程序可以使一些 web 页面、博客、wiki、基于 web 的日历应用或商业网站。

Flask 属于微框架（*micro-framework*）这一类别，微架构通常是很小的不依赖于外部库的框架。这既有优点也有缺点，优点是框架很轻量，更新时依赖少，并且专注安全方面的 bug，缺点是，你不得不自己做更多的工作，或通过添加插件增加自己的依赖列表。Flask 的依赖如下：

- Werkzeug (http://werkzeug.pocoo.org/) 一个 WSGI 工具包
- jinja2 (http://jinja.pocoo.org/) 模板引擎

维基百科 WSGI 的介绍：

> Web服务器网关接口（Python Web Server Gateway Interface，缩写为WSGI）是为Python (https://zh.wikipedia.org/wiki/Python)语言定义的Web服务器 (https://zh.wikipedia.org/wiki/%E7%B6%B2%E9%A0%81%E4%BC%BA%E6%9C%8D%E5%99%A8)和Web应用程序 (https://zh.wikipedia.org/wiki/%E7%BD%91%E7%BB%9C%E5%BA%94%E7%94%A8%E7%A8%8B%E5%BA%8F)或框架 (https://zh.wikipedia.org/wiki/Web%E5%BA%94%E7%94%A8%E6%A1%86%E6%9E%B6)之间的一种简单而通用的接口 (https://zh.wikipedia.org/wiki/%E4%BB%8B%E9%9D%A2_(%E7%A8%8B%E5%BC%8F%E8%A8%AD%E8%A8%88))。自从WSGI被开发出来以后，许多其它语言中也出现了类似接口。

# 2.2 什么是模板引擎？

你搭建过一个网站吗？你面对过保持网站风格一致的问题吗，你不得不写多次相同的文本吗？你有没有试图改变这种网站的风格？

如果你的网站只包含几个网页，改变网站风格会花费你一些时间，这确实可行。尽管如此，如果你有许多页面（比如在你商店里的售卖物品列表），这个任务便很艰巨。

使用模板你可以设置你的页面的基本布局，并提及哪个元素将发生变化。这种方式可以定义您的网页头部并在您的网站的所有页面使它保持一致，如果你需要改变网页头部，你只需要更新一个地方。

使用模板引擎创建/更新/维护你的应用会节约你很多时间。

# 2.3 "Hello World" 应用

我们将使用 flask 完成一个非常基础的应用。

- 安装 flask

```
$ sudo pip3 install flask
```

- 创建项目结构

```
$ mkdir -p hello_flask/{templates,static}
```

这是你的 web 应用的基本结构：

```
$ tree hello_flask/
hello_flask
|-- static
`-- templates

2 directories, 0 files
```

`templates` 文件夹是存放模板的地方， `static` 文件夹存放 web 应用所需的静态文件（images, css, javascript）。

- 创建应用文件

```
$ cd hello_flask
$ vim hello_flask.py
```

hello_flask.py 文件里编写如下代码：

```
#!/usr/bin/env python3

import flask


# Create the application.
APP = flask.Flask(__name__)


@APP.route('/')
def index():
    """ 显示可在 '/' 访问的 index 页面
    """
    return flask.render_template('index.html')


if __name__ == '__main__':
    APP.debug=True
    APP.run()
```

- 创建模板文件 `index.html`

```
$ vim templates/index.html
```

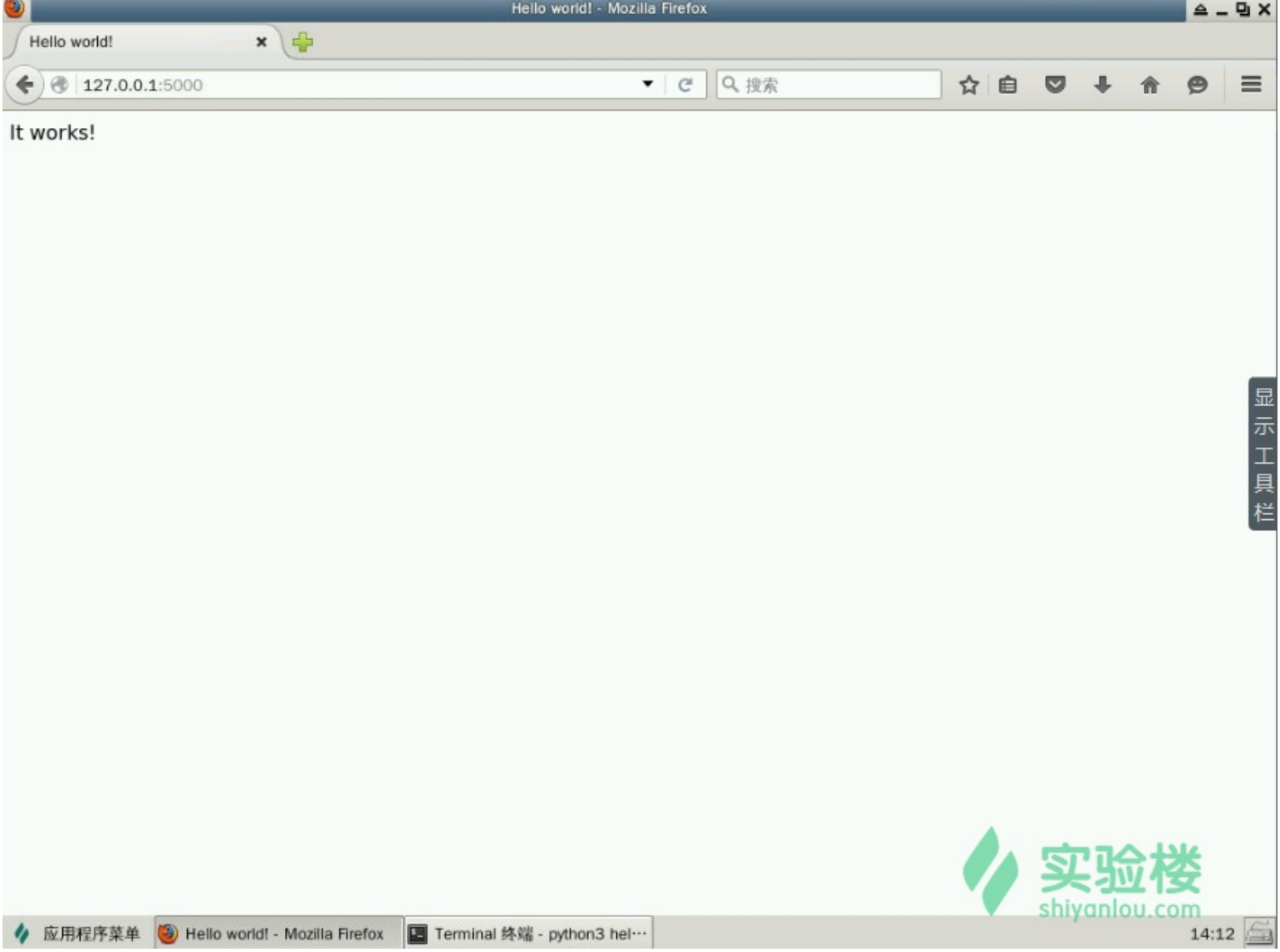
index.html 文件内容如下：

```
<!DOCTYPE html>
<html lang='en'>
<head>
  <meta charset="utf-8" />
  <title>Hello world!</title>
  <link type="text/css" rel="stylesheet"
        href="{{ url_for('static',
              filename='hello.css')}}" />
</head>
<body>

It works!

</body>
</html>
```

- 运行 flask 应用程序

```
$ python3 hello_flask.py
```

访问 http://127.0.0.1:5000/ (http://127.0.0.1:5000/)，这应该只是显示黑字白底的 "It works!" 文本，如下图：




# 2.4 Flask 中使用参数

在本节中我们将要看到如何根据用户使用的 URL 返回网页。

为此我们更新 hello_flask.py 文件。

- 在 hello_flask.py 文件中添加以下条目

```
@APP.route('/hello/<name>/')
def hello(name):
    """ Displays the page greats who ever comes to visit it.
    """
    return flask.render_template('hello.html', name=name)
```

- 创建下面这个模板 hello.html

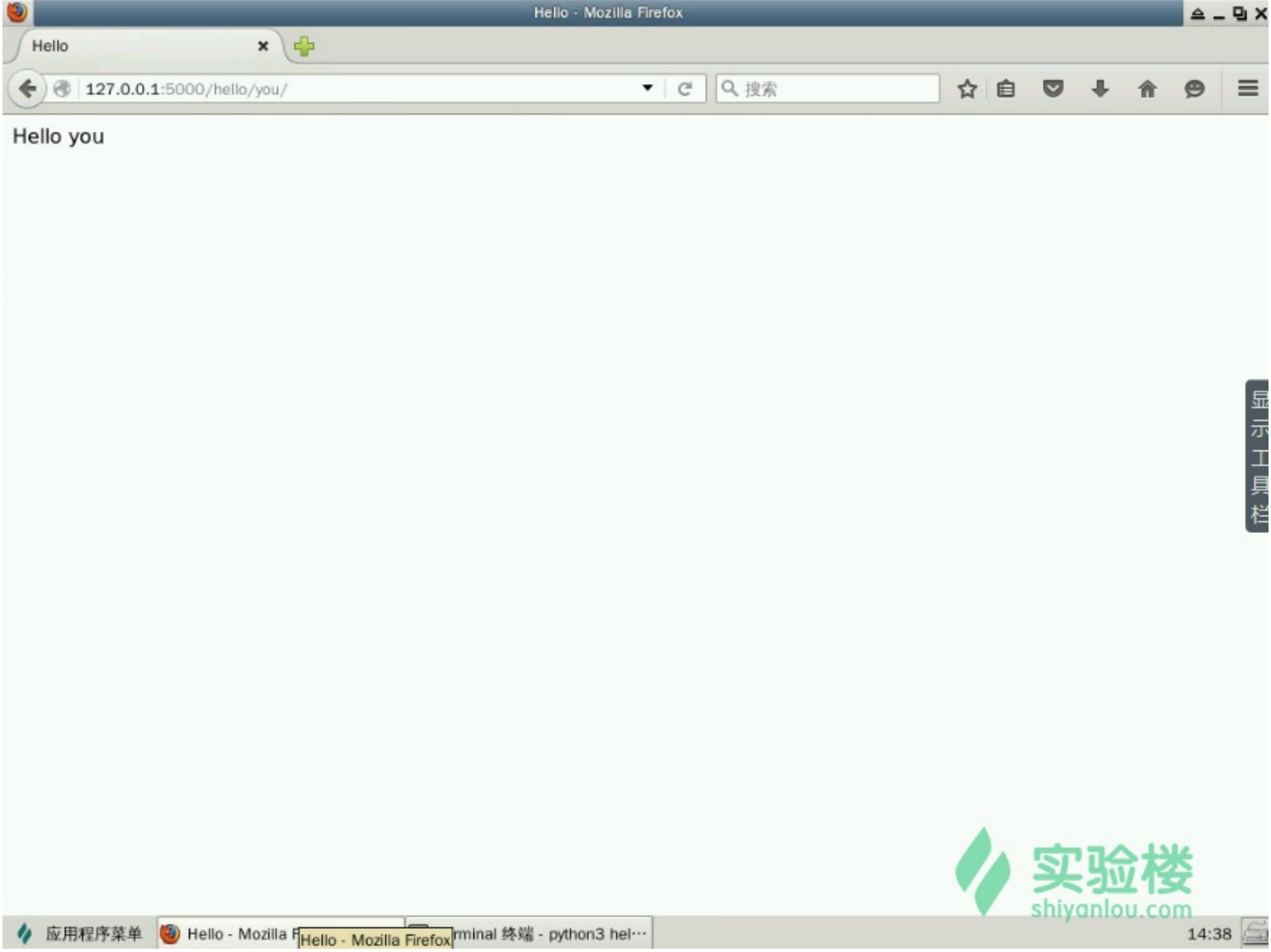
```
<!DOCTYPE html>
<html lang='en'>
<head>
    <meta charset="utf-8" />
    <title>Hello</title>
    <link type="text/css" rel="stylesheet"
          href="{{ url_for('static',
               filename='hello.css')}}" />
</head>
<body>

      Hello {{name}}

</body>
</html>
```

- 运行 flask 应用

```
$ python3 hello_flask.py
```

访问 http://127.0.0.1:5000/ (http://127.0.0.1:5000/) ，这应该只是显示黑字白底的 "It works!" 文本。

访问http://127.0.0.1:5000/hello/you (http://127.0.0.1:5000/hello/you)，这应该返回文本 "Hello you"，见下图：




无论你在 URL 中 `/hello/` 后填写的什么，都会出现在返回的网页中。

这是你第一次使用模板，我们在 hello_flask.py 中建立了 `name` 变量（参见 hello 函数的 return 行）。通过语法 `{{name}}` ，name 变量之后在页面中显示其自身。

# 2.5 额外工作

## 2.5.1 使用模板

目前，对于每一个页面我们都创建了一个模板，其实这是不好的做法，我们应该做的是创建一个主模板并且在每个页面使用它。

- 创建模板文件 master.html。

```
<!DOCTYPE html>
<html lang='en'>
<head>
  <meta charset="utf-8" />
  <title>{% block title %}{% endblock %} - Hello Flask!</title>
  <link type="text/css" rel="stylesheet"
       href="{{ url_for('static',
                        filename='hello.css')}}" />
</head>
<body>

{% block body %}{% endblock %}

</body>
</html>
```

- 调整模板 index.html。

```
{% extends "master.html" %}

{% block title %}Home{% endblock %}

{% block body %}
It works!
{% endblock %}
```

正如你所看到的，在 master.html 模板中我们定义了两部分，名为 `title` 和 `body` 的 `blocks`。

在模板 index.html 中，我们声明这个模板扩展自 master.html 模板，然后我们定义了内容来放在这两个部分中（blocks）。在第一个 block `title` 中，我们放置了 `Home` 单词，在第二个 block `body` 中我们定义了我们想要在页面的 body 中有的东西。

- 作为练习，更改其他模板 hello.html，同样要使用 master.html。
- 在 hello 页面添加首页链接。

调整模板 hello.html，添加到首页的链接。

```
<a href="{{ url_for('index') }}"><button>Home</button></a>
```

- 作为你的任务，在首页添加到 hello 页面的链接。

# 三、总结

本实验中我们了解了微框架、WSGI、模板引擎等概念，学习使用 Flask 做一个 web 应用，在这个 web 应用中，我们使用了模板。而用户以正确的不同 URL访问服务器时，服务器返回不同的网页。最后还给大家留了一个小任务，希望大家能完成。

这里只介绍了 Flask 很少的一部分，有兴趣的可以学习 Flask 的官方文档 (http://flask.pocoo.org/)。



上一节：项目结构 (/courses/596/labs/2053/document)