理论盘点：Web 客户端与服务端

目录

Web 客户端和服务端主要功能

- Web 客户端：一般为我们使用的 Chrome、Firefox 等浏览器，主要用于解决用户与服务器交互的问题，可以将用户的请求提交到服务器，也能够解释服务器的图片、代码并展示给用户

- Web 服务端：一般用来接收 Web 客户端的请求，并向更上游的数据库请求数据，通过中间件处理完数据后，以特定的形式返回给浏览器，进而展示给用户

Web 服务器需要处理的主要数据

- HTTP 协议头数据
- HTTP 包体
- HTTP 返回状态码

Web 服务器需要处理的主要数据

HTTP 协议头又包含了请求方式和请求内容两部分：

1. 最常用的请求方式为 Get 和 Post 两种
2. 请求内容主要以 URL 为主，其他如浏览器版本、主机名等信息

   服务器也会根据请求内容进行处理

使用 http.server 模块构建简单 Web 服务器

* Python 内置了简单 Web 服务器 —— http.server 模块
* 在需要发布的目录直接运行 python3 –m http.server

总结

1. Web 客户端实现网站内容的展示功能，Web 服务端处理 Web 客户端的请求并将处理完成的数据返回给客户端

2. http.server 模块可以实现简单的 Web 服务端，并可以用于 Web 程序开发和测试

课后作业

请你使用 http.server 模块，将目录 /tmp 通过 HTTP 协议发布到 8080 端口，并通过浏览器验证是否发布成功。

理论盘点：MVC 模型是什么？

目录

Web 框架的作用

没有 Web 框架之前：

1. 用户登陆–查询数据库–返回页面

2. 匿名用户–查询数据库–返回其他页面

…

Web 框架的作用

完全手动编写 Web 服务端逻辑带来的问题：

1. 有大量重复逻辑
2. 耦合紧密，一处修改可能会引发另一处 Bug
3. 依赖关系不清晰

Web 框架的作用

- Web 框架就像盖楼时的结构，确保 Web 服务器代码

  有层次、松耦合、不累赘

- 常见的 Web 框架包括 MVC 和 MTV 两种模式，它们的本质相同

MVC 模型是什么

MVC 模型将 Web 后端代码分为三个层次：

- M：Model 模型层，实现业务对象和数据库对象之间的映射
- V：View 视图层，负责业务逻辑和用户交互
- C：Controller 控制层，实现用户请求到视图层的调用

Django 的 MTV 和 MVC 模型的区别

- Python 最知名的框架 Django 实现了 MTV 模型
- 除了和 MVC 模型定义上不同外，解耦和依赖解决方面均采用相同的处理逻辑

Django 的 MTV 和 MVC 模型的区别

M：Model 模型层

T：Template 页面模版

V：View 视图层

Django 的 MTV 和 MVC 模型的区别

Django 的数据处理顺序为：

用户请求 → URL 控制器 → view 视图 →[Model 模型] → Template 模版 → 返回 HTML

总结

1. MVC 模型实现了数据分层和松耦合的后端服务器

2. 著名的 Django 开发框架使用了和 MVC 功能相同，分层相似的 MTV 模型，理解 MVC 和 MTV 模型是开发成熟的 Web 后端程序的基础

课后作业

请你使用pip安装并启动 Django 框架， 并使用浏览器访问 Django 监听的端口。 根据网页的返回信息，验证Django是否被成功安装和启动。

如何使用 Django 搭建简单的 Web 服务器?

目录

请求和响应处理

- 创建项目

```
django-admin startproject myproject
```

- 创建应用

```
python3 manage.py startapp testapp
```

- 运行Django

```
python3 manage.py runserver 8888
```

请求和响应处理

- 处理 URL

```
urlpatterns = [
    path('admin/', admin.site.urls),
]
```

模型

```
class Choice(models.Model):
    question = models.ForeignKey(Question, on_delete=models.CASCADE)
    choice_text = models.CharField(max_length=200)
    votes = models.IntegerField(default=0)

class Question(models.Model):
    question_text = models.CharField(max_length=200)
    pub_date = models.DateTimeField('date published')
```

视图和模板

```
{% if latest_question_list %}
    <ul>
    {% for question in latest_question_list %}
        <li><a href="/polls/{{ question.id }}/">{{ question.question_text }}</a></li>
    {% endfor %}
    </ul>
{% else %}
    <p>No polls are available.</p>
{% endif %}
```

总结

1. Django Web 框架包含了 URL 处理、模型、视图、模板四个主要部分

2. URL 处理负责用户请求到视图的映射，视图用于处理用户逻辑，
   模型用于映射 Python 对象和数据对象，模版用于展示特定信息的样式给用户

课后作业

请你将所在地区的气温、天气存入数据库，并利用 Django 框架，

将数据库中的天气信息进行展示。

如何使用 Django-admin 实现文章上传?

目录

添加模型

- 文章对象模型应当包含：文章标题、文章类别、文章作者、发布时间和文章内容
- 字段类型参考：

https://docs.djangoproject.com/zh-hans/4.0/ref/models/fields/

* 切记要注册 model 到管理后台，否则模型无法在管理后台显示

从管理后台上传数据

管理界面地址：http://127.0.0.1/admin

优化显示字段

- 列表页默认不显示全部字段，通过以下方式显示：

```
class blogAdmin(admin.ModelAdmin):

    list_display = (字段名称)

# 注册管理类到管理界面：

admin.site.register(blog, blogAdmin)
```

总结

1. Django-admin 可以轻松实现文章的发布上传功能
2. Django-admin 默认界面不够友好，需按照业务特点进行界面优化

课后作业

请你根据这节课内容，通过 Django-admin 实现 Blog 的上传功能。

如何使用 Django 实现文章发布?

目录

制作模板

- 模板通过 templates 文件夹里的 HTML 展示给用户
- 模板除内容外，还可以展示样式以及 JS 等前端脚本实现的逻辑

修改视图

- 根据上一讲创建的 Blog 模型，创建展示的视图

* 注意：创建视图后，需要修改 urls.py 映射用户请求 URL 到指定的视图

修改视图

- 修改视图后，如果以开发模式运行 Django，刷新浏览器后可查看最新的页面信息
- 如果以生产模式运行 Django，需重启 Django 进程

总结

1. 文章发布是将数据库中的指定字段通过模板展示给用户

2. 视图需绑定 URL，并由 Django 重新加载后才能正常访问

课后作业

请你将 Blog 的文章标题、分类、时间和内容展示到网页上。

THANKS