**Flask2.0 输出**

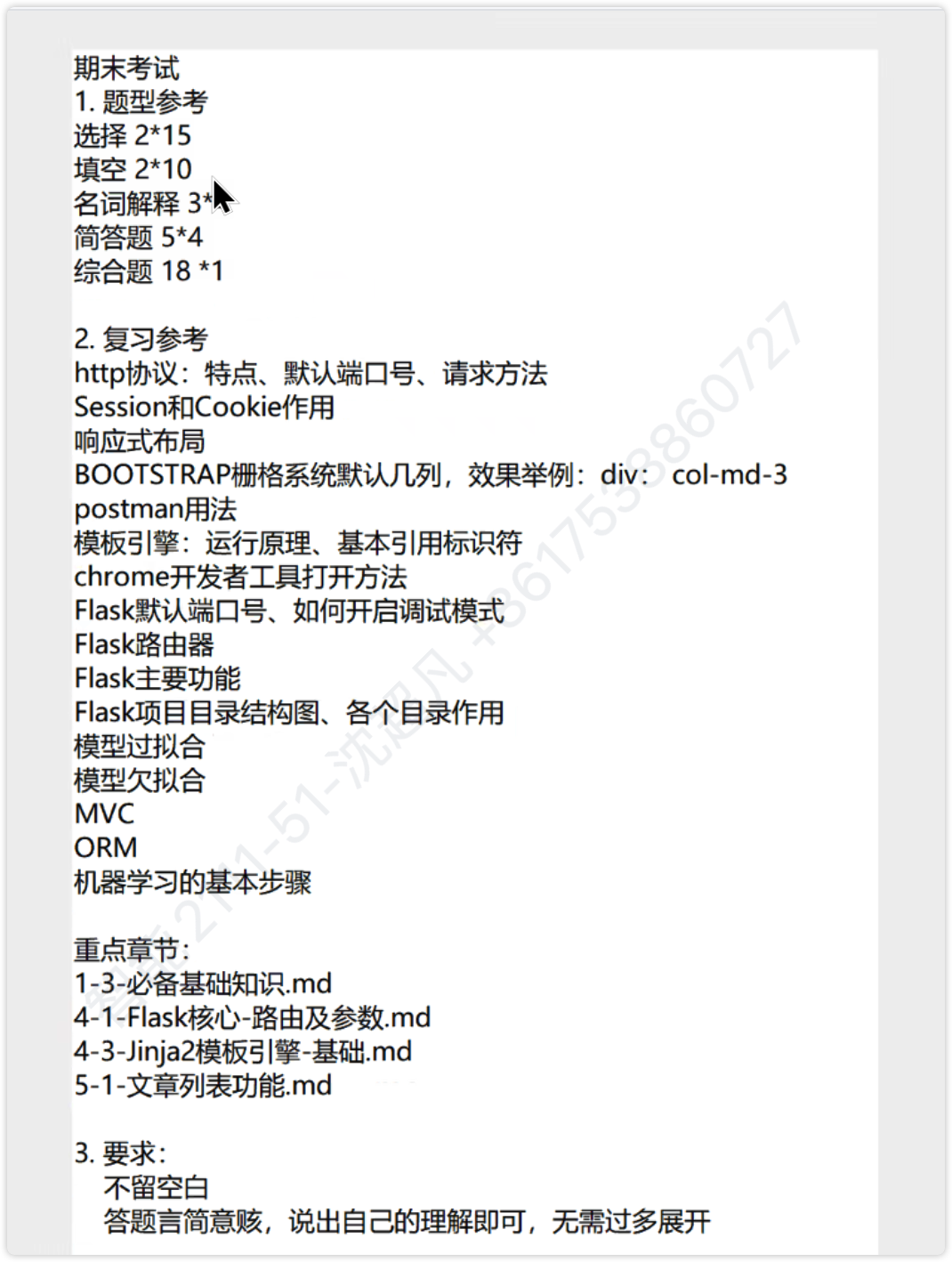
Author: 智能2111 沈超凡

Description: 修复一些已知问题

Date: 2023-02-11 00:51:29

LastEditors: jarrycf 1354602179@qq.com

LastEditTime: 2023-02-20 21:45:07



**选择 15\*2**

1. 下面代码会把布局分成几列？

|  |
| --- |
| HTML div：col-md-3 |

4列

#每一列有原本的三行，所以是4列

1. jQuery使用什么来进行单个或批量元素的选择？

(“＃id”)和(“.class”)

1. jQuery使用那两个函数获取或修改元素或表单的内容？

.val()或.text()

1. jQuery的哪两个通过Ajax方式发送请求并处理服务器的响应结果？

.ajax()和.post()

1. jQuery的哪三个函数用于按照元素的层次进行定位？

.parent()

.siblings()

.children()

1. jQuery的什么函数用于修改HTML元素的样式？

.css()

1. jQuery的什么函数可以将HTML内容动态添加到另外一个元素中？

append()

1. 栅格系统从什么类开始，还有哪些类？

container

.container-fluid

.row

.col-\*

1. 下面5个值分别在什么大小的屏幕上使用，第一个值是什么意思？

|  |
| --- |
| HTML <!-- col-xl-3: --> <!-- col-lg-3: --> <!-- col-md-6: --> <!-- col-sm-6: --> <!-- col-12: --> |

在超大屏幕及以上的屏幕上该列占3个栅格单元宽度

表示在超大屏幕（的浏览器窗口上）显示为3列宽度（ container总宽度的25%）

大屏幕

中等屏幕

小屏幕

超小屏幕

1. 栅格系统种哪两个类表示在中等屏幕以下浏览器上隐藏该元素？

d-none

d-md-block

1. 手工配置静态资源和模板文件名的参数分别是什么？

static\_folder

template\_folder

1. 判断

Controller是指：通过@app.route装饰器与被装饰的函数绑定，用于对请求进行后台处理，这部分代码

并将id转换为int类型使用的是<int:id>参数

postman可以将该请求保存下来，随时可以按需调试

对对对

1. 对资源进行下列操作是什么风格？

1、 GET：获取资源

2、POST：新增资源

3、PUT：修改资源

4、DELETE：删除资源

RESTful风格

1. 注册蓝图的代码是什么？

app.register\_blueprint()

**填空 10\*2**

1. http的默认端口号是多少？

80

1. 栅格系统默认多少列

12

1. 开发者的三种打开方式

F12 设置-更多工具 右击检查

1. flask默认端口号

5000

1. 如何开启调试模式？

debug=True

1. cookie保存在哪？

客户端

1. flask的模板引擎叫什么

Jinja2

**名词解释 4\*3**

响应式布局？

以相同的代码适配不同的屏幕

响应式布局是一种web设计技术，它通过CSS媒体查询（media query）来实现。它使得你只需要编写一套代码，就可以适配不同屏幕大小的设备如手机，平板，电脑，还能帮助网站获得一致的外观和功能。

什么是模型欠拟合和过拟合，如何解决？

欠拟合：训练集和验证集上表现都不好

1. 通过数据增强：如对图片进行平移、旋转、翻转生成新的训练数据
2. 通过增加模型复杂性：如增加网络层数、节点数、卷积核大小
3. 通过SGD、MSGD减少每次的训练量
4. 通过动量法（Momentum）手动调整学习率
5. 通过Adam自适应学习率
6. 迁移学习

过拟合：训练集上表现好，测试集上表现不好

1. L1、L2正则化
2. 早停（Early Stopping）
3. 随机失活（Dropout）
4. 批归一化（Batch Normalization）
5. 集成学习，如Bagging、Boosting等
6. 交叉验证

MVC 是什么?

MVC（Model-View-Controller）模型-视图-控制器，是web框架的一种目录结构设计模式。

它将应用程序分为三部分

1. model：处理数据和业务之间的逻辑，通常用于与数据库交互。
2. view：处理用户请求和响应，通常用于呈现页面的HTML/CSS/JavaScript代码。
3. controller：处理请求和响应之间的逻辑，通常用于调用相应的模型和视图来完成任务。

MVC目录结构可以使代码更加清晰，易于扩展和维护。

ORM 是什么？

ORM（Object-Relational Mapping）对象关系映射，是一种将对象模型和关系型数据库模型进行映射的技术。

ORM允许你通过使用Python类和方法来定义和操作数据库表，而无需直接使用SQL语句。flask中这样的工具包有SQLAlchemy和Peewee等

优点：  
提高开发效率

提高可读性

缺点：

ORM需要对底层数据库进行二次封装，因此一些情况下性能会有所下降

ORM并不能完全覆盖所有的数据库操作。

**简答题 4\*5**

简述http协议的三个特点？

无状态：服务器无法记住客户端的各种状态，所以需要通过Session和Cookie来解决。

单通道：只能客户端主动向服务器发起请求，反过来则不行，所以需要WebSocket技术实现双向通信。

非加密：整个HTTP协议的传输过程是完全明文传输，所以需要HTTPS协议进行加密传输。

简述HTTP协议4种主流请求类型的功能和作用？

GET请求：用于访问服务器某个资料，也可以通过URL地址的查询字符串来向服务器提交参数。

POST请求：用于向服务器提交一段数据。

PUT请求：用于更新服务器某个资源。

DELETE请求：用于删除服务器某个资源。

为什么需要session和cookie，他们的作用又分别是什么？  
由于HTTP协议属于无状态协议，服务器无法记住客户端的各种状态，所以需要通过session和cookie来解决。

Session使用一个唯一的session ID来跟踪用户的会话状态，而Cookie用于记录用户的行为、偏好设置等。

cookie在客户端发出请求时创建，保存在客户端中

session在务器端接收到请求时创建，保存在服务器中

注：

为了在web应用中标识用户身份和跟踪用户行为。

Session是服务器端的一种状态管理技术，通常用于存储用户登录信息、购物车信息等敏感数据。

cookie是存储在客户端浏览器中的一小段数据，通常用于存储用户偏好设置、广告跟踪、用户ID等数据。

由于HTTP协议属于无状态协议，服务器无法记住客户端的各种状态，所以需要通过session和cookie来解决。

Session使用一个唯一的session ID来跟踪用户的会话状态。当用户第一次访问网站时，服务器会创建一个session，并将session ID存储在Cookie中返回给浏览器。之后当用户再次访问网站时，浏览器都会将这个session ID传递回服务器，从而保持用户的会话状态。

Cookie用于记录用户的行为、偏好设置等。当服务器向客户端发送HTTP的响应报文时，可以在响应头中设置Set-Cookie字段来创建一个Cookie，并将Cookie的值存储在浏览器中。当用户再次访问网站时，浏览器会自动将Cookie发送给服务器，从而实现跟踪和个性化服务。

flask主要功能？

1. 路由分发
2. 视图处理
3. 模板引擎
4. 数据库操作
5. web表单处理
6. 安全性保护
7. Session和Cookie
8. 文件上传

机器学习的基本步骤是什么？

数据采集 数据预处理 特征工程 模型训练和预测 模型评估 模型优化 模型应用

注：

1. 数据采集：选择合适的数据源，收集数据样本，确定特征值和目标值
2. 数据预处理：清洗、转换、规范化、缺失值处理等，提高数据质量
3. 特征工程：选择、构造、降维等，提取有效的特征，减少冗余和噪声
4. 模型选择：根据问题类型和数据特点，选择合适的机器学习算法，如监督学习、非监督学习或强化学习等。
5. 模型训练和预测：利用训练集对模型进行参数优化，利用测试集或新数据对模型进行预测或分类
6. 模型评估：使用各种指标和方法，如准确率、召回率、F1分数、ROC曲线等，评估模型的性能和泛化能力。
7. 模型优化：根据评估结果，对模型进行调参、优化等处理，以提高模型的性能。
8. 模型应用：将模型应用于实际问题中，并进行预测和决策。

具体实现方式和步骤可能因具体的应用场景而有所不同

**综合题 1\*18**

回答问题，实现蜗牛博客的文章列表功能？

画出Flask项目的一级目录并说明作用？

static目录：用于保存静态资源，如缩略图、头像、文章上传的图片等。

templates：用于保存模板页面，供Flask渲染。

controller：用于保存控制层的所有源代码。

model：用于保存模型层的所有源代码，主要封装数据库操作。

common：用于保存一些公共组件，如邮件短信发送、验证码生成、数据库连接等通用功能。  
无法归为controller和model的代码都可以保存在common目录中。

main.py：用于定义程序入口及拦截器等，是Flask的入口程序。

模型引擎的页面渲染流程有哪三步？

把HTML文件当成普通文本文件进行解析

找到HTML文件中的模板引擎标识，用预先定义好的规则进行替换和数据填充

将填充内容作为一个长字符串返回给前端作为响应正文

1. 解析HTML文件，查找模板引擎标识；
2. 进行替换和数据填充；
3. 将填充后的内容返回给前端作为响应正文。

模板引擎的三种基本引用标识符的用法是什么？

{% ... %} 用于循环或判断语句

{{ ... }} 用于表达式值的引用

{# ... #} 用于模板引擎的注释  
# 如果<!-- -->注释中存在模板引擎的语法，将被执行。此时需要使用{# #}来注释

根据下列操作说明完成填空（共 7 空）

使用模板引擎对index.html页面进行数据填充。由于Flask只给模板页面传递了result的查询结果变量，所以需要在模板页面中利用Jinja2的语法规则进行数据填充。

|  |
| --- |
| Python  <-- 继承 base.html 基础模板--> \_\_1\_\_  <-- 填写一个占位符，用于后续拓展 --> \_\_2\_\_   <!-- 中部区域布局 --> <div class="container" style="margin-top: 20px;">  <div class="row">  <div class="col-sm-9 col-12" style="padding: 0 10px;" id="left">  <!-- 轮播图组件应用，除了修改图片路径外，其他内容可不修改 -->  <div id="carouselExampleIndicators" class="col-12 carousel slide"  data-ride="carousel" style="padding: 0px">  <!-- 轮播图代码此处省略 -->  </div>   <!-- 循环遍历result变量的值，并以article 和  users 进行对应位置的数据填充 -->  \_\_3\_\_  <div class="col-12 row article-list">  <div class="col-sm-3 col-3 thumb d-none d-sm-block">  <img src="/thumb/{{article.thumbnail}}" class="img-fluid" style="width: 210px; height: 125px; border-radius: 5px" />  </div>    <-- 使用Bootstrap框架的栅格系统，大屏幕占3个列，小屏幕占12列 容器名为detail -->  \_\_4\_\_    <div class="title"><a href="/article/{{article.articleid}}"> {{article.headline}}</a></div>  <div class="info"> 作者：{{users.nickname}}&nbsp;&nbsp;&nbsp; 类别：{{article.type}}&nbsp;&nbsp;&nbsp;  日期：{{article.createtime}}&nbsp;&nbsp;&nbsp; 阅读：{{article.readcount}} 次&nbsp;&nbsp;&nbsp; 消耗积分：{{article.credit}} 分</div>  <div class="intro">      <!-- 利用Jinja2的过滤器完成对文章内容摘要article.content的处理：不转义、过滤变量中的HTML标签、截断字符串长度为80，并在截断处添加省略号-->  \_\_8\_\_  </div>  </div>  </div>    <-- 结束for循环-->  \_\_5\_\_   <!-- 分页功能暂时不做处理 -->  <div class="col-12 paginate">  <a href="#">上一页</a>&nbsp;&nbsp;  <a href="#">1</a>&nbsp;&nbsp;  <a href="#">2</a>&nbsp;&nbsp;  <a href="#">3</a>&nbsp;&nbsp;  <a href="#">4</a>&nbsp;&nbsp;  <a href="#">5</a>&nbsp;&nbsp;  <a href="#">下一页</a>  </div>  </div>    # 将 "side.html" 的模板文件插入到当前模板  \_\_6\_\_  </div> </div>  <-- 结束子模板中重写或扩展的块 --> \_\_7\_\_ |

{% extends 'base.html' %}

{% block content %}

{% for article, users in results %}

<div class='col-sm-3 col-xs-12 detail '>

{{article.content | safe | striptags | truncate(80, True)}}

{% endfor %}

{% include 'side.html' %}

{% endblock %}

填写下列两空

|  |
| --- |
| Python # for循环 {% for article, nickname in result %} # 定义一个12列宽的行（超小屏幕及以上的屏幕上该列占12个栅格单元） css样式名为article-list \_1\_  # 在小屏幕及以上的屏幕上该列占3个栅格单元 超小屏幕及以上的屏幕上该列占3个栅格单元 小屏幕以下设备上隐藏，而在小屏幕及以上设备上显示。 样式为thumb类  \_2\_  <img src="/thumb/{{article.thumbnail}}" class="img-fluid"/>  </div>  # 小于等于sm屏幕尺寸时，所占用的列数为9 在超小型屏幕上，所占用的列数为12  \_3\_  <div class="title"><a href="/article/{{article.articleid}}"> {{article.headline}}</a></div>  <!-- 填充作者姓名时直接填充nickname -->    <div class="intro">  <!-- 利用Jinja2的过滤器完成对文章内容摘要的处理 -->  {{article.content | safe | striptags | truncate(80, True)}}  </div>  </div> </div> {% endfor %} |

<div class="col-12 row article-list">

<div class="col-sm-3 col-3 thumb d-none d-sm-block">

class="col-sm-9 col-12"

flask 开启 debug，指定端口80打开web应用程序的代码是什么？

app.run(port=80, debug = True)

flask路由器是用来干嘛的？

1. Flask路由器用于将HTTP请求映射到对应的处理函数，
2. Flask路由器可以定义处理函数接受的请求类型，如GET、POST、PUT、DELETE等。
3. Flask路由器可以获取请求体的数据或URL中的查询参数，并将其传递给对应的处理函数进行处理。

注：

将HTTP请求映射到对应的处理函数，解决URL地址定义问题

它可以实现

1. 定义服务器接口的URL地址，从 / 根目录开始。
2. 定义接收前端数据的请求类型，如GET、POST、PUT、DELETE等
3. 获取请求体的数据或URL中的查询参数

实现一个名为article\_read(id) 的路由，路由url为/articel/，指定任意字符串参数，请求方式为get

|  |
| --- |
| Python @app.route('/article/<id>', methods = ['GET']) def article\_read(id):  pass |

如何用postman是干嘛的，用来测试一个接口，大致的步骤有哪些？

postman是一个测试http协议的工具

1. 在postman地址栏中输入API的URL，创建一个新的请求
2. 选择请求的HTTP方法，如GET、POST、PUT、DELETE方法之一
3. 添加请求的参数，如在请求体中添加JSON数据或在URL参数中添加查询参数
4. 在请求头中添加必要的标头，如授权标头或接受标头等
5. 发送请求并查看响应报文是否包含预期的信息
6. 重复上述步骤，直到对API的测试结果满意为止

**补充（都每考到）**

什么是bootstrap栅格系统？

Bootstrap栅格系统是一种用于网页设计和布局的前端框架，它基于12列的栅格系统，可以轻松地将网页内容布局成多列，并通过设定不同的屏幕尺寸对应的栅格大小，实现响应式布局。Bootstrap栅格系统的主要作用是方便开发人员进行网页设计和布局，使得网页在不同的设备上显示效果更加协调和美观。同时，Bootstrap栅格系统也提供了一些常用的CSS样式和JavaScript插件，如导航条、表格、表单、模态框等，方便开发人员快速搭建网页。

**其他（都没考到）**

BBBBC

1. Flask中的蓝图可以用来做什么？  
   A) 定义路由  
   B) 分离应用功能  
   C) 定义模型
2. 下面哪个命令可以在Flask中启用调试模式？  
   A) app.debug = True  
   B) app.run(debug=True)  
   C) app.start\_debug()
3. Flask中用来处理异常的装饰器是什么？  
   A) @app.route('/error')  
   B) @app.errorhandler(Exception)  
   C) @app.exception(Exception)
4. Flask中用来接收POST请求的方法是什么？  
   A) @app.get()  
   B) @app.post()  
   C) @app.route()
5. 在Flask中，可以使用什么来访问Web应用程序中的静态文件？  
   A) @app.static\_folder  
   B) @app.static\_file()  
   C) @app.route('/static')

ABAAC

1. 在Flask中，使用什么方法可以从URL参数中获取值？  
   A) request.args.get()  
   B) request.form.get()  
   C) request.values.get()
2. 下面哪个Flask扩展可以用来生成表单？  
   A) Flask-Forms  
   B) Flask-WTForms  
   C) Flask-InputForms
3. 在Flask中，用来设置cookie的方法是什么？  
   A) response.set\_cookie()  
   B) response.add\_cookie()  
   C) response.create\_cookie()
4. 在Flask中，使用什么方法可以重定向到另一个URL？  
   A) redirect()  
   B) refresh()  
   C) forward()
5. Flask中用来实现用户认证的扩展是什么？  
   A) Flask-Auth  
   B) Flask-Security  
   C) Flask-Login

AAACB

1. 在Flask中，用来实现ORM的库是什么？  
   A) SQLAlchemy  
   B) Flask-ORM  
   C) Flask-SQL
2. 下面哪个Flask扩展可以用来发送邮件？  
   A) Flask-Mail  
   B) Flask-Email  
   C) Flask-SMTP
3. 在Flask中，用来处理表单验证的库是什么？  
   A) Flask-Input  
   B) Flask-Check  
   C) Flask-FormValidation
4. 在Flask中，用来管理会话的库是什么？  
   A) Flask-Sessions  
   B) Flask-Cookies  
   C) Flask-Session
5. 在Flask中，用来处理跨域访问的库是什么？  
   A) Flask-Ajax  
   B) Flask-CORS  
   C) Flask-Origin

AAABA

1. Flask 是哪个编程语言的 Web 框架？  
   A) Python  
   B) JavaScript  
   C) Ruby  
   D) PHP
2. Flask 使用哪个 ORM 工具来管理数据库？  
   A) SQLAlchemy  
   B) Django ORM  
   C) Peewee  
   D) Pony ORM
3. 在 Flask 中，通过哪个装饰器可以定义一个路由？  
   A) @app.route()  
   B) @app.run()  
   C) @app.url()  
   D) @app.view()
4. 在 Flask 中，如何获取 HTTP 请求中的表单数据？  
   A) request.args  
   B) request.form  
   C) request.json  
   D) request.headers
5. 在 Flask 中，如何获取 HTTP 请求中的查询参数？  
   A) request.args  
   B) request.form  
   C) request.json  
   D) request.headers

AACCA

1. 在 Flask 中，如何将一个变量传递给模板渲染引擎？  
   A) 通过参数传递  
   B) 通过 URL 传递  
   C) 通过 Session 传递  
   D) 通过 Cookie 传递
2. 在 Flask 中，如何返回一个 JSON 响应？  
   A) jsonify()  
   B) render\_template()  
   C) make\_response()  
   D) send\_file()
3. 在 Flask 中，如何设置一个 Cookie？  
   A) response.cookie()  
   B) request.cookie()  
   C) response.set\_cookie()  
   D) request.get\_cookie()
4. 在 Flask 中，如何获取当前用户的 IP 地址？  
   A) request.remote\_ip  
   B) request.ip\_address  
   C) request.client\_ip  
   D) request.user\_agent
5. 在 Flask 中，如何处理文件上传？  
   A) 使用表单提交文件  
   B) 通过 AJAX 上传文件  
   C) 使用 FTP 上传文件  
   D) 使用 SFTP 上传文件

DAAAB

1. 在 Flask 中，如何创建一个定时任务？  
   A) 使用 threading 模块  
   B) 使用 multiprocessing 模块  
   C) 使用 Timer 类  
   D) 使用 cron 表达式
2. 在 Flask 中，如何使用 Blueprint 来组织路由？  
   A) 定义一个 Blueprint 对象，将其与 Flask 应用绑定  
   B) 使用 Flask 对象直接定义路由  
   C) 使用 app.route() 装饰器定义路由  
   D) 使用 Flask 对象的 register\_blueprint() 方法注册路由
3. 在 Flask 中，如何创建一个 WebSocket 服务？  
   A) 使用 Flask-SocketIO 扩展  
   B) 使用 Flask-WebSocket 扩展  
   C) 使用 Flask-WSGI 扩展  
   D) 使用 Flask-Socket 扩展
4. 在 Flask 中，如何实现用户认证和授权？  
   A) 使用 Flask-Login 扩展  
   B) 使用 Flask-Auth 扩展  
   C) 使用 Flask-Session 扩展  
   D) 使用 Flask-Security 扩展
5. 在 Flask 中，哪个函数用于将指定函数绑定到特定的 URL 路径上？

A. url\_for()  
B. route()  
C. bind()  
D. bind\_route()

AADD

1. 在 Flask 中，使用哪个函数可以将 Python 对象转换为 JSON 字符串？

A. jsonify()  
B. to\_json()  
C. dump()  
D. serialize()

1. 在 Flask 中，使用哪个方法可以从 HTTP 请求中获取表单数据？

A. request.form  
B. request.json  
C. request.args  
D. request.data

1. 在 Flask 中，使用哪个模块来连接 SQLite 数据库？

A. flask.db  
B. flask.sql  
C. flask.sqlite  
D. flask\_sqlalchemy

1. 在 Flask 中，使用哪个装饰器可以指定 HTTP 请求方法？

A. @app.route()  
B. @app.method()  
C. @app.get()  
D. @app.post()

微型 @app.route Jinja2 render\_template request.args

1. Flask是一种 \_\_\_\_\_\_ 框架。
2. Flask中，可以使用 \_\_\_\_\_\_ 装饰器将函数绑定到指定的URL。
3. 在Flask中，可以使用\_\_\_\_\_\_\_\_\_实现模板的渲染。
4. 在Flask中，使用\_\_\_\_\_\_\_\_\_可以将变量传递给模板。
5. 在Flask中，使用\_\_\_\_\_\_\_\_\_可以获取HTTP请求中的参数。

SQLAlchemy set\_cookie Flask-Login Flask-RESTful Ajax

1. Flask中，可以使用\_\_\_\_\_\_\_\_\_对数据库进行操作。
2. 在Flask中，使用\_\_\_\_\_\_\_\_\_可以设置Cookie。
3. 在Flask中，使用\_\_\_\_\_\_\_\_\_可以实现用户认证。
4. 在Flask中，使用\_\_\_\_\_\_\_\_\_可以实现API接口。
5. 在Flask中，可以使用\_\_\_\_\_\_\_\_\_实现与前端的交互。

WTForms redirect request.headers.get Flask-Uploads methods

1. 在Flask中，使用\_\_\_\_\_\_\_\_\_可以对表单进行验证。
2. Flask中，可以使用\_\_\_\_\_\_\_\_\_将请求重定向到指定的URL。
3. 在Flask中，使用\_\_\_\_\_\_\_\_\_可以从HTTP请求头中获取指定的参数。
4. 在Flask中，使用\_\_\_\_\_\_\_\_\_可以实现文件上传功能。
5. 在Flask中，使用\_\_\_\_\_\_\_\_\_可以设置路由的请求方法。

errorhandler Flask-Principal Flask-Cache Flask-Rewrite config

1. Flask中，可以使用\_\_\_\_\_\_\_\_\_将指定的函数注册为错误处理函数。
2. 在Flask中，使用\_\_\_\_\_\_\_\_\_可以获取当前用户的信息。
3. 在Flask中，使用\_\_\_\_\_\_\_\_\_可以实现缓存功能
4. 在Flask中，使用\_\_\_\_\_\_\_\_\_可以对URL进行重写
5. 在Flask中，使用\_\_\_\_\_\_\_\_\_可以获取当前应用的配置信息。

Flask-Mail Flask-Pagination Flask-Scheduler Flask-CORS before\_request

1. 在Flask中，使用\_\_\_\_\_\_\_\_\_可以实现邮件发送功能。
2. 在Flask中，可以使用\_\_\_\_\_\_\_\_\_实现分页功能。
3. 在Flask中，使用\_\_\_\_\_\_\_\_\_可以实现定时任务功能。
4. 在Flask中，使用\_\_\_\_\_\_\_\_\_可以实现跨域访问。
5. 在Flask中，使用\_\_\_\_\_\_\_\_\_可以对请求进行预处理。

Flask-Cache Flask-Logging Flask-Session Blueprint request.path

1. Flask中，可以使用\_\_\_\_\_\_\_\_\_将数据存储到缓存中。
2. 在Flask中，使用\_\_\_\_\_\_\_\_\_可以实现日志记录。
3. 在Flask中，使用\_\_\_\_\_\_\_\_\_可以实现会话管理。
4. 在Flask中，可以使用\_\_\_\_\_\_\_\_\_实现蓝图功能。
5. 在Flask中，使用\_\_\_\_\_\_\_\_\_可以获取请求的路径。

什么是http，实现过程是怎么样的？

HTTP（Hypertext Transfer Protocol）超文本传输协议。它建立在TCP/IP协议之上，它定义了客户端和服务器之间交互的规则和格式，以便实现Web页面的请求和响应。

通讯过程分为5步

1. 客户端发起请求：客户端将请求打包成 HTTP 的请求报文，报文包括三部分：起始行、请求头、请求体，再采用 TCP 发送给服务器
2. 服务器处理请求：服务器接收到请求报文后按 HTTP 协议进行解析
3. 服务器返回响应：服务器根据解析后获知的客户端请求进行逻辑执行，并将执行后的结果打包成 HTTP 的响应报文，报文包括三部分：起始行，响应头，响应体，其中状态码在起始行中，再采用刚才的 TCP 连接发送给客户端
4. 客户端处理响应：客户端接收到响应报文后按HTTP协议进行解析获取结果
5. 客户端渲染页面：如果响应是一个HTML页面，客户端会解析HTML代码，下载页面所需的资源（如图片、脚本等），并将页面呈现给用户。

BOOTSTRAP栅格系统是什么？

它是一种基于12个栏目网格系统的响应式布局解决方案

将页面布局划分为不同的水平栏，每个栏宽度占用栅格系统中的一定比例，从而实现灵活的响应式布局。例如，你可以将一行分为12个栏，一个栏宽度占据1个栅格，也可以将一行分为6个栏，每个栏宽度占据2个栅格。

通过Bootstrap提供的组件和样式即可快速构建响应式页面

postman是用来干什么的，它的用法是什么？

Postman是一款API测试工具

它可以实现下列4中任务

1. 测试API：测试API的不同请求方法，例如GET、POST、PUT和DELETE等，以验证API的功能和正确性。
2. 调试API：查看API请求和响应的详细信息，例如请求头、响应头、请求参数和响应体等，以便更快速地排查问题
3. 模拟API：在API开发的早期阶段，你可以使用Postman来模拟API的行为和响应，以快速实现API的设计和测试。
4. 管理API：Postman提供了集合和环境等功能，可以方便地管理多个API的请求和响应，并且可以与其他开发人员共享API集合，以提高协作效率。

主要的语法规则有：

* {{ ... }}：表示一个变量占位符，用于将变量的值插入到模板中。
* {% ... %}：表示一个控制流语句，用于控制模板的渲染方式。
* {# ... #}：表示一个注释语句，用于添加注释到模板中。
* {{ ...|filter }}：表示一个变量过滤器，用于对变量的值进行过滤处理。
* {% include 'template.html' %}：表示一个模板引用语句，用于将一个子模板引用到当前模板中。

如：

然后使用语法规则实现if-else条件语句的三种代码是什么？

{% if user %}

{% else %}

{% endif %}

**考后总结**

就上面这点内容很简单

选择题考了

* Elements :查找网页源代码HTML中的任一元素,手动修改任一元素的属性和样式且能实时在浏览器里面得到反馈。
* Console :记录开发者开发过程中的日志信息，且可以作为与JS进行交互的命令行Shell。
* Sources :断点调试JS。
* Network :从发起网页页面请求Request后分析HTTP请求后得到的各个请求资源信息（包括状态、资源类型、大小、所用时间等），可以根据这个进行网络性能优化。
* Timeline :记录并分析在网站的生命周期内所发生的各类事件，以此可以提高网页的运行时间的性能。

最后一题考了发送邮件在前端和后端的执行过程以及邮件没有出现在邮箱的问题排查方法。

在写postman测试ip时，要写填写请求体的数据是什么，我忘了