

第1章 绪论

讲授：任胜兵
中南大学 计算机学院



中南大學

1.6 软件工程工具与环境



中南大學
CENTRAL SOUTH UNIVERSITY

內容摘要

软件工程工具

集成开发环境

AI辅助软件工程

软件开发框架Flask

01 软件工程工具



中南大學
CENTRAL SOUTH UNIVERSITY

工欲善其事，必先利其器



对一个待开发的系统，先考虑采用何种方法（看待系统的立场、观点等），然后
再考虑采用何种工具（提高开发质量和效率）。

软件工具是指为支持计算机软
件的开发、维护及有关工作而
研制的程序系统。

使用软件工具的目的是降低软
件开发和维护的成本，提高软
件产品的生产效率和质量。

软件开发工具



用于软件开发过程的
各种开发活动的工具

软件维护工具



用于对代码及其有关文档
进行各种维护活动的工具

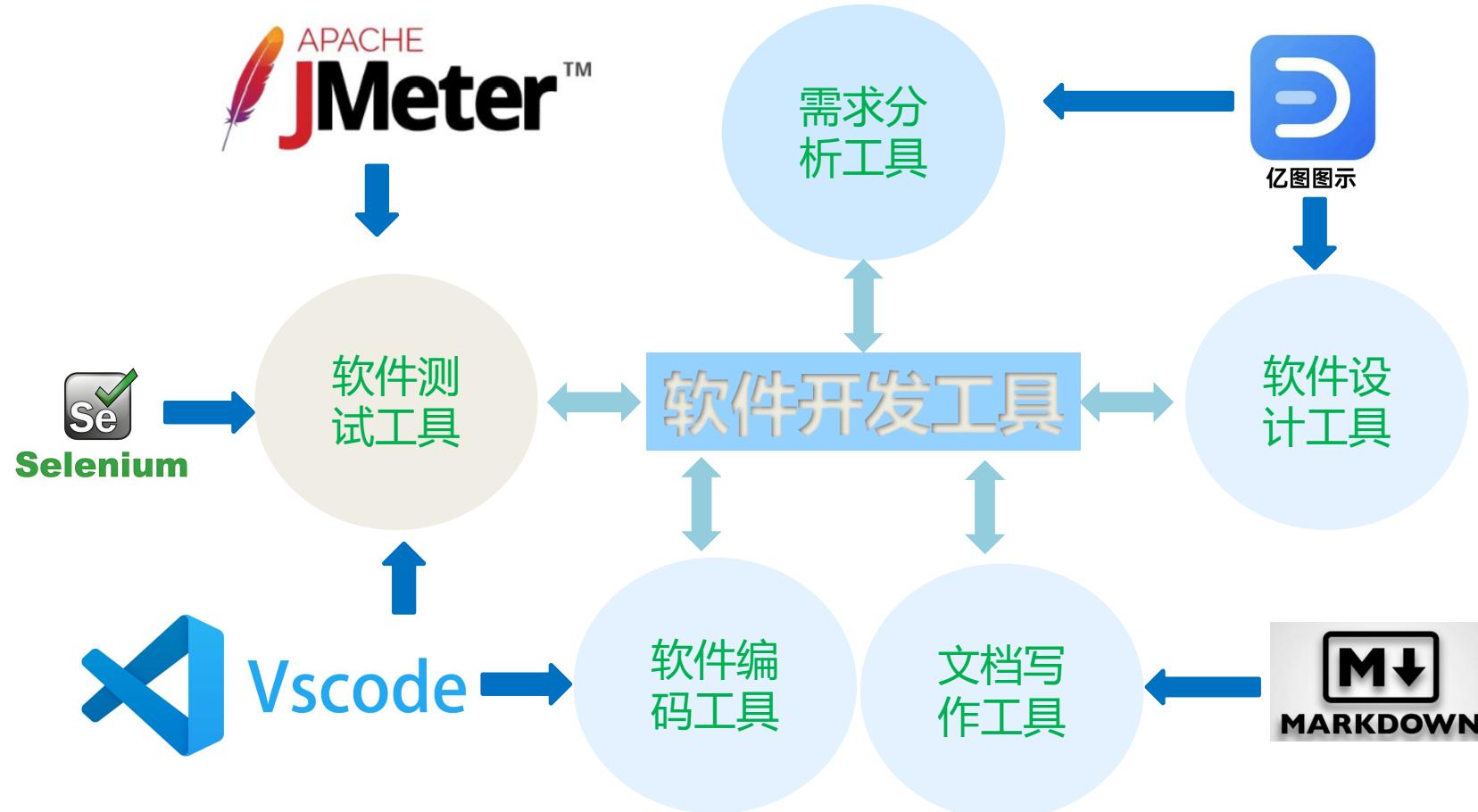
项目管理工具



用于项目各种管理和支持活动的工具

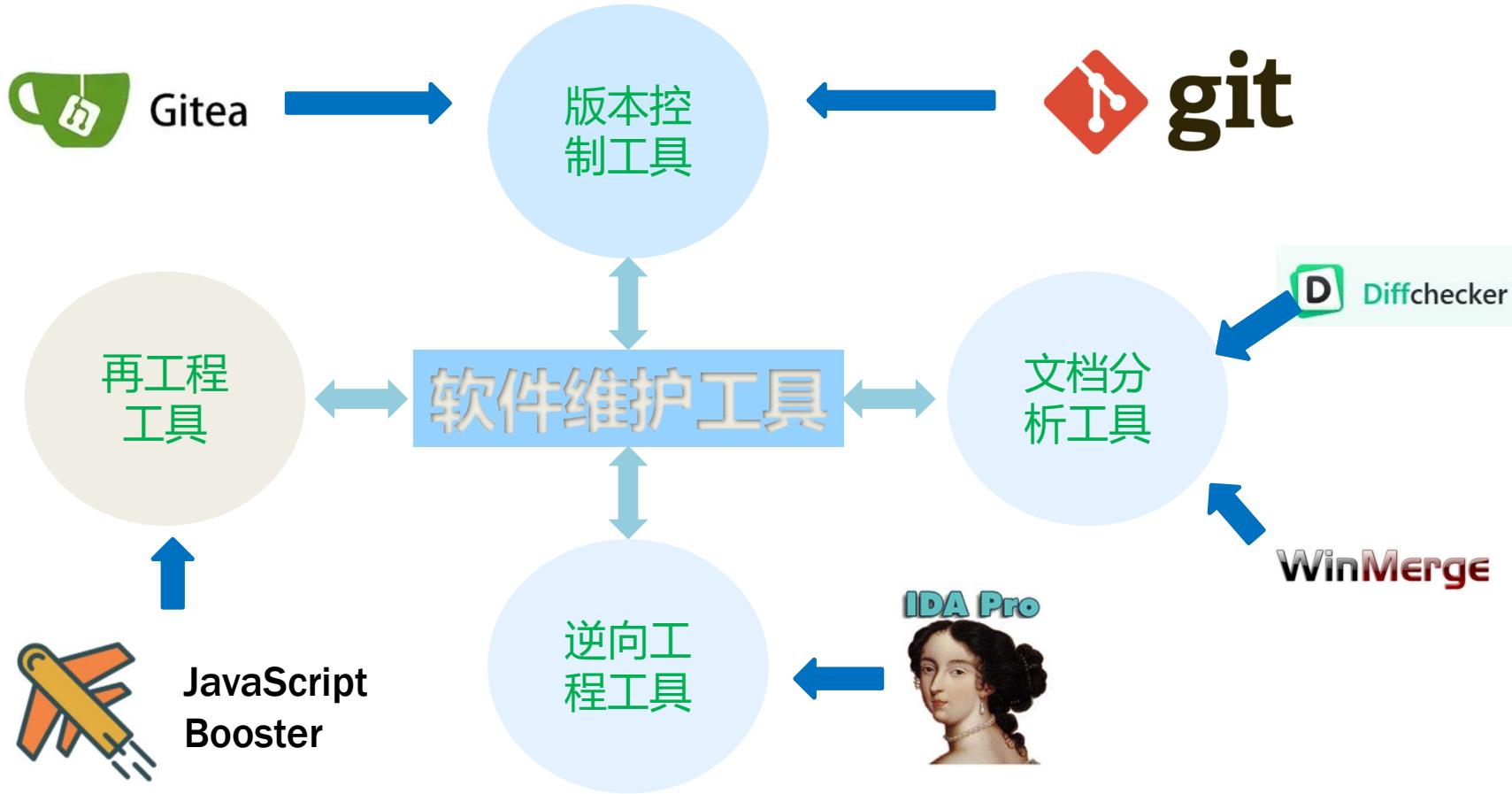
使用工具的目的是提高效率和质量。

01 软件工程工具



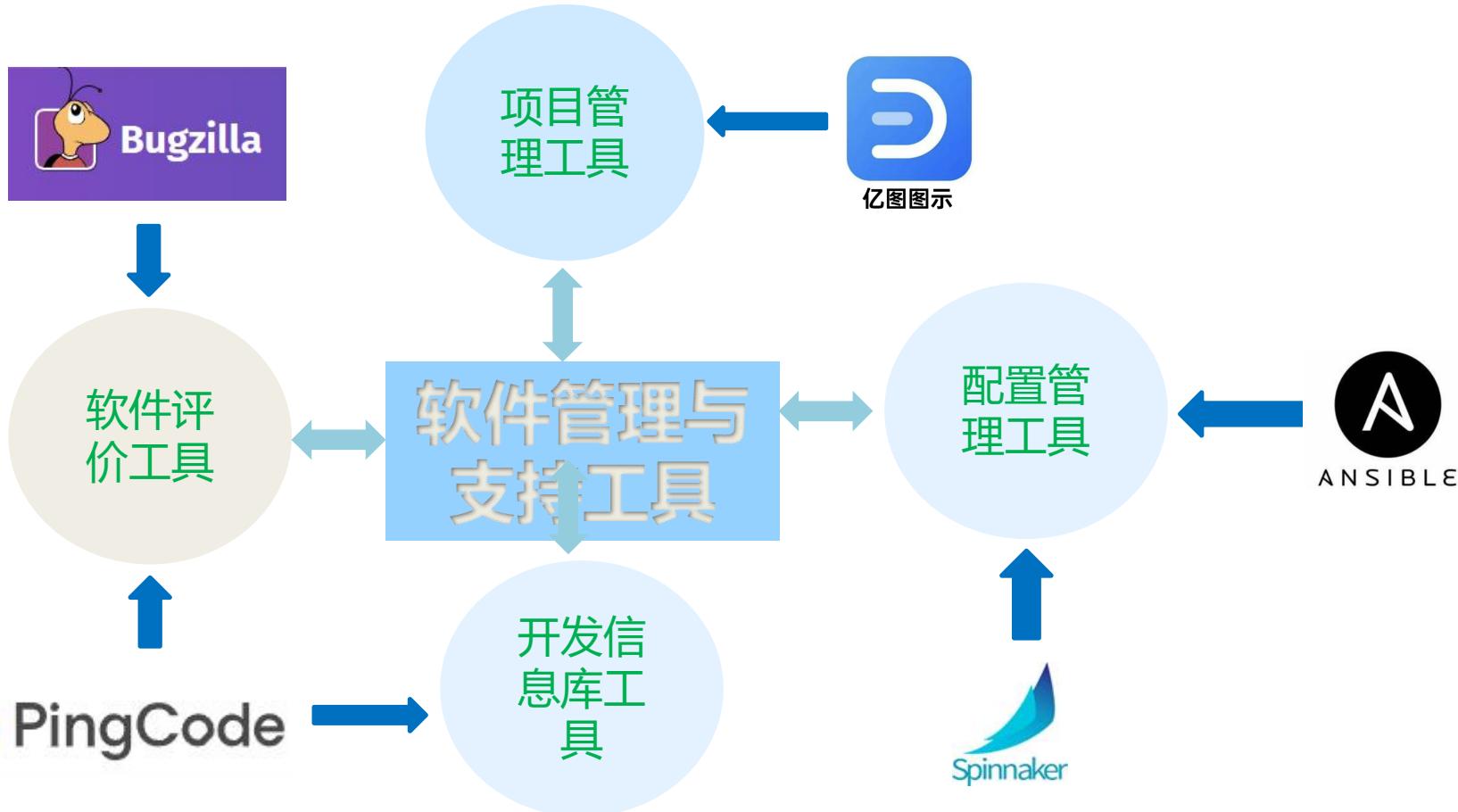
软件工程师必须能灵活实用工具

01 软件工程工具



软件工程师必须能灵活实用工具

01 软件工程工具



软件工程师必须能灵活实用工具

软件工程工具存在的问题

- 一般情况下一种软件工具只支持一种活动。
(软件开发和维护过程中进行的活动较多)

- 工具界面不统一，工具内部无联系，工具切换由人工操作。 (对大型软件的开发和维护的支持能力受限)

解决办法：工具集成化



CASE将软件工具和开发方法集成

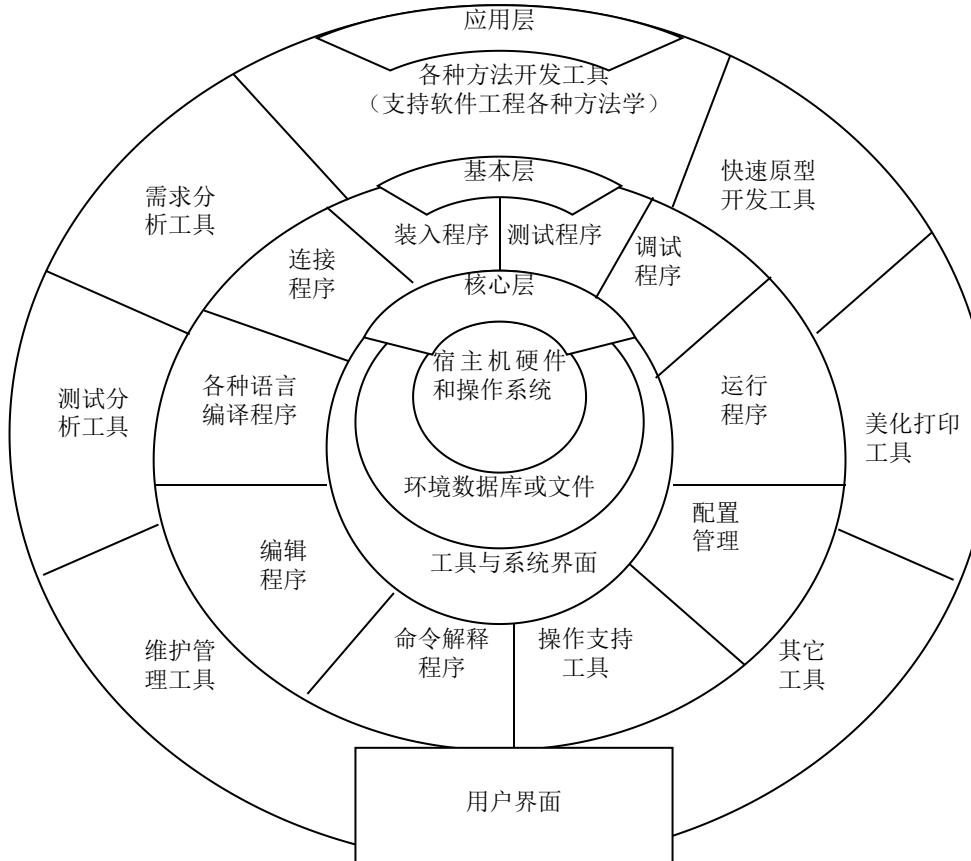


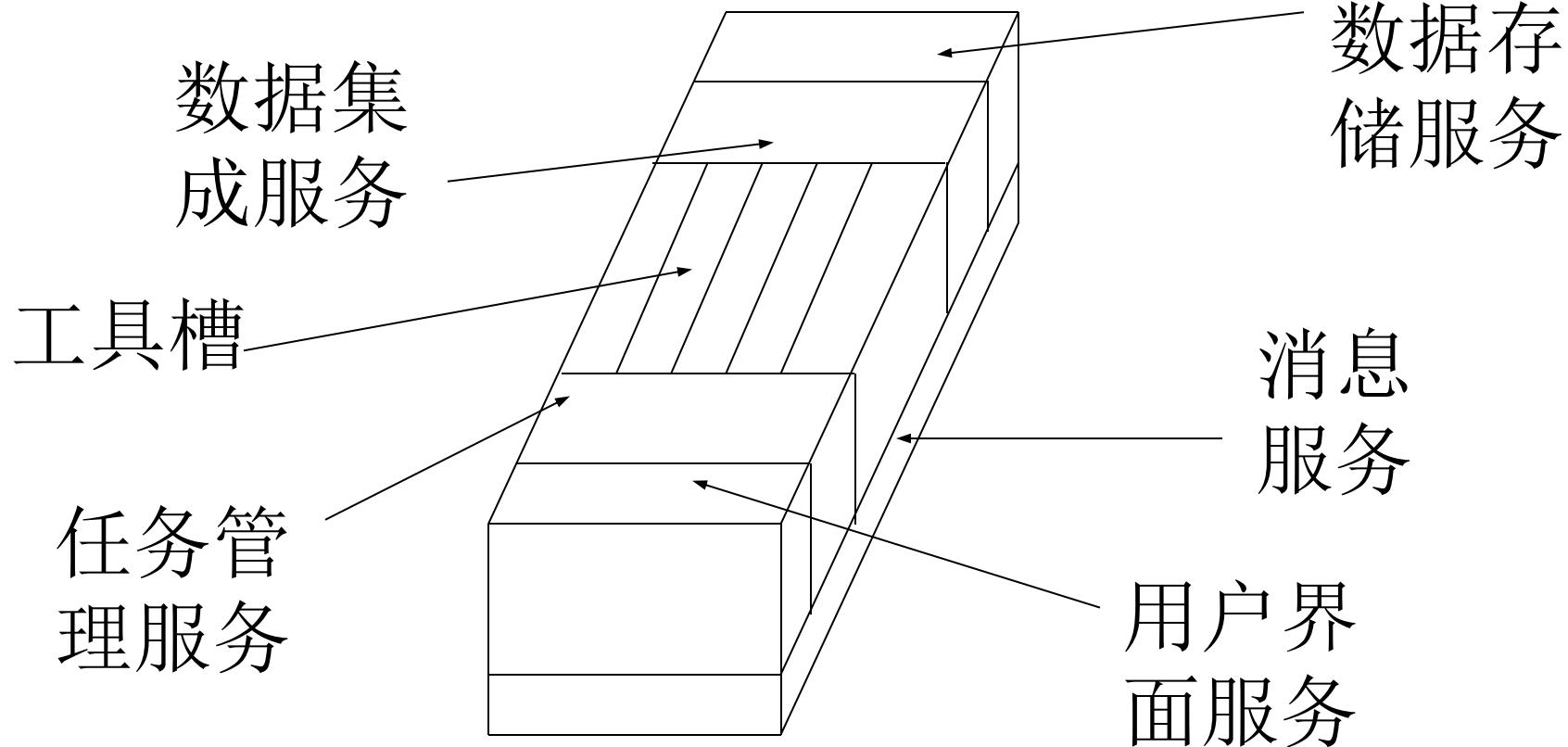
- 由软件**工具集**和环境**集成机制**构成。
- **软件工具集**用以支持软件开发的相关过程、活动和任务；（支持某种**开发方法**）
- **环境集成机制**为工具集成和软件开发、维护和管理提供统一的支持。



集成化项目支持环境 (Integrated Project Support Environment, IPSE)

UNIX
Shell语言





北大：青鸟系统



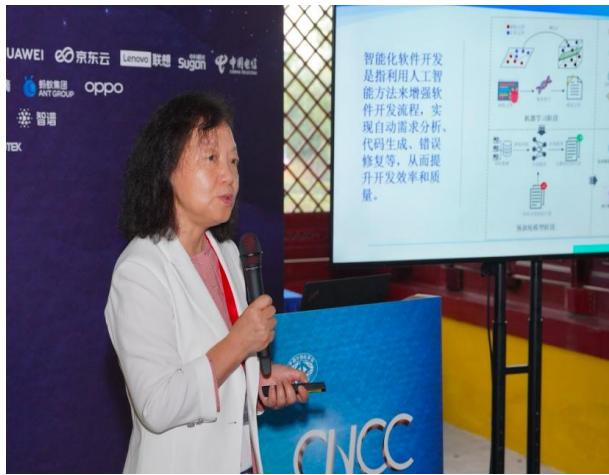
软件工程+AI（智上加智）

03 AI辅助软件工程



中南大學
CENTRAL SOUTH UNIVERSITY

百度AI: AI辅助软件开发是指利用**生成式人工智能技术**来辅助软件工程师进行软件开发的过程。它包括**代码生成、代码优化、错误检测、代码审查、自动化测试**等多个方面，旨在提高**开发效率、降低成本、提升软件质量**。

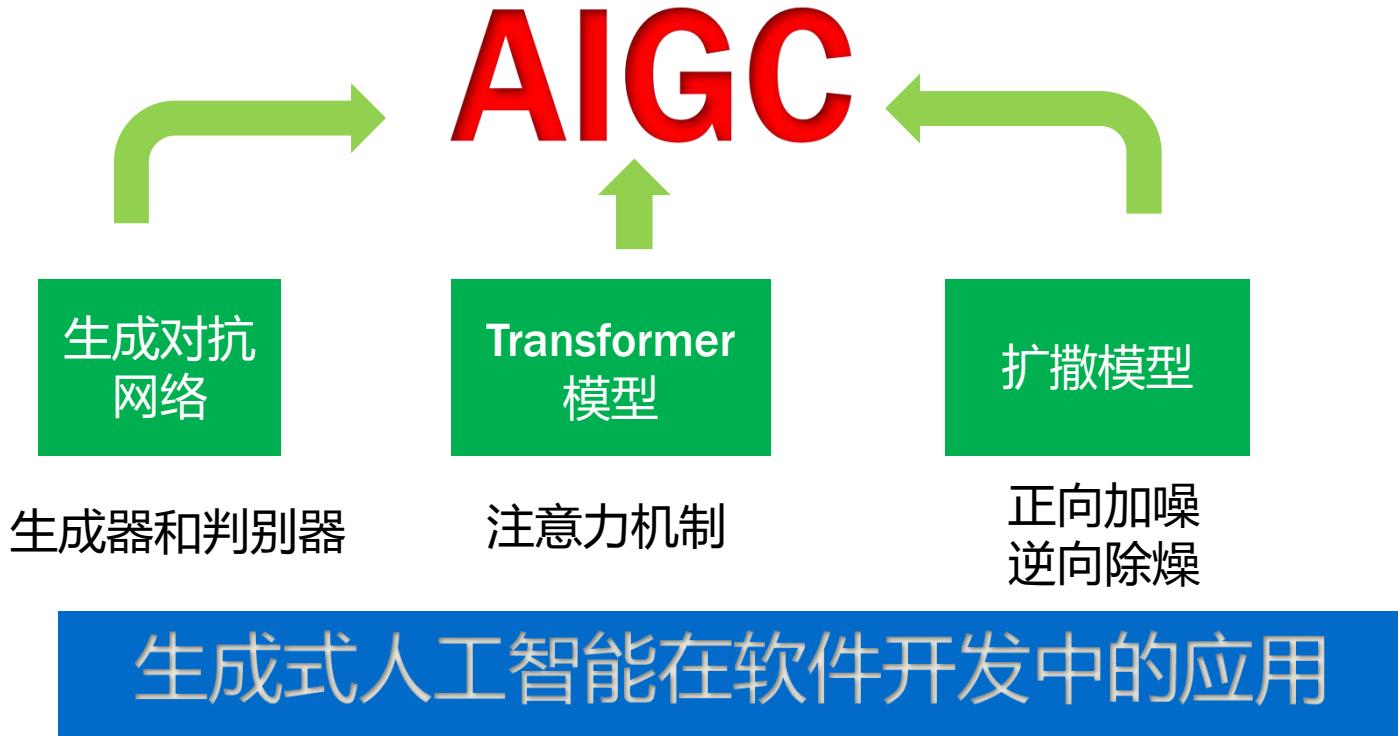


2024年度中国计算机大会—“大模型如何使能软件工程



生成式人工智能在软件开发中的应用

AIGC: 全称为Artificial Intelligence Generated Content, 是指利用人工智能技术生成的内容。





生成式人工智能（AIGC）赋能软件工程师：

- **提升效率**：AIGC能够自动化执行重复性高且耗时的任务，如代码生成和自动化测试，从而让开发者将更多时间投入到解决更复杂的问题上。
- **提高质量**：通过AIGC的错误检测与修复功能，开发者可以快速识别并修复代码中的错误，减少人为疏漏，提高代码质量。
- **促进创新**：AIGC能使软件工程师加快探索新的编程模式和解决方案，推动技术进步。
- **降低难度**：AIGC工具通过提供代码示例和自动化的编程指导，降低了学习编程的难度，使更多人能够参与到软件开发中。

生成式人工智能在软件开发中的应用

03 AI辅助软件工程



中南大學
CENTRAL SOUTH UNIVERSITY



<https://xiaohuanxiong.com/code>



<https://www.marscode.cn/workbench>



我是 DeepSeek，很高兴见到你！

我可以帮你写代码、读文件、写作各种创意内容，请把你的任务交给我吧~

<https://chat.deepseek.com/>



腾讯云 AI 代码助手

<https://copilot.tencent.com/>

生成式人工智能在软件开发中的应用



请帮我用 HTML 生成一个五子棋游戏，所有代码都保存在一个 HTML 中。



已深度思考（用时 34 秒） ^

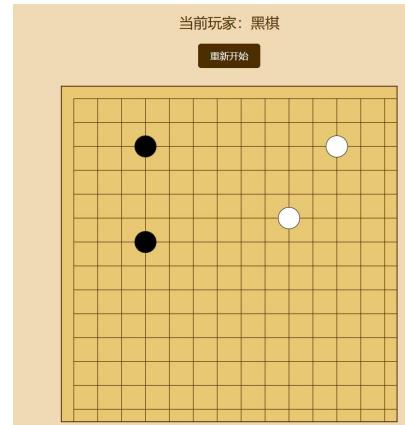
好的，用户让我帮忙用HTML生成一个五子棋游戏，并且所有代码都保存在一个HTML文件中。首先，我需要明确用户的需求。五子棋是一个经典的棋盘游戏，通常需要棋盘、黑白棋子交替下子、判断胜负等功能。用户可能希望这个游戏能够在浏览器中运行，不需要后端，所以所有逻辑都要用前端技术实现，包括HTML、CSS和JavaScript。

以下是使用纯HTML+CSS+JavaScript实现的五子棋游戏完整代码，保存为单个HTML文件即可运行：

```
html
<!DOCTYPE html>
<html>
<head>
<title>五子棋游戏</title>
<style>
body {
  display: flex;
  flex-direction: column;
  align-items: center;
  background: #f0f0f5;
  font-family: Arial, sans-serif;
}

#board {
  background: #e8c873;
  border: 2px solid #4d2e00;
  margin: 20px;
}

.cell {
  width: 40px;
  height: 40px;
}
```



生成式人工智能在软件开发中的应用

Web



获取API Key的步骤:

- 进入百度千帆ModelBuilder平台，注册并登录 (需要刷脸认证)
- 在百度智能云管理中心页面->API Key->创建API Key这里创建API key，权限配置中的服务选择千帆ModelBuilder；资源选择所有资源，随后点击确定即可复制API key

<https://console.bce.baidu.com/iam/#/iam/apikey/list>

API使用规则

API



首先pip3 install openai安装Open AI SDK

```
api_key='bce-v3/ALTAK-KZke*****/f1d6ee*****'
base_url='https://qianfan.baidubce.com/v2'
model='deepeseek-v3'                      #DeepSeek-V3模型
model='deepeseek-r1'                      #DeepSeek-R1模型
还有其他很多模型:
https://cloud.baidu.com/doc/WENXINWORKSHOP/s/Fm2vrveyu#%E6
%94%AF%E6%8C%81%E6%A8%A1%E5%9E%8B%E5%88%97%E8%A
1%A8
```

生成式人工智能在软件开发中的应用

03 AI辅助软件工程



```
from openai import OpenAI

client = OpenAI(
    base_url="https://qianfan.baidubce.com/v2",
    api_key="bce-v3/ALTAK-ZnUVdIUJec9QHUEiBpyTE/c0005085929e4eed9109db2b588e2e19cf03af65")

messages=[
    {
        "role": "user",
        "content": "请解释下面这段代码的逻辑，并说明完成了什么功能：\n```\n// weight数组的大小 就是物品个数\nfor(int i = 1; i < weight.size(); i++) { // 遍历物品\n    for(int j = 0; j <= bagweight; j++) { // 遍历背包容量\n        if (j < weight[i]) dp[i][j] = dp[i - 1][j];\n        else dp[i][j] = max(dp[i - 1][j], dp[i - 1][j - weight[i]] + value[i]);\n    }\n}\n```\n"
    }
]

response = client.chat.completions.create(
    model="deepseek-r1",
    messages=messages
)

reasoning_content = response.choices[0].message.reasoning_content
content = response.choices[0].message.content

print(reasoning_content)
print(content)
```

生成式人工智能在软件开发中的应用

03 AI辅助软件工程



中南大學
CENTRAL SOUTH UNIVERSITY

嗯，我现在得仔细看看这段代码，然后试着理解它到底在做什么。首先，这段代码看起来像是动态规划的问题，可能跟背包有关。对，因为变量名里有`bagweight`，还有`weight`数组和`value`数组，所以应该是经典的背包问题吧。

....
这段代码实现了动态规划算法来解决0-1背包问题，但存在逻辑错误。它的目标是计算在给定背包容量下，能装入物品的最大总价值。具体分析如下：

```
### **功能解释**  
1. **问题目标**  
....  
### **修正后的代码逻辑**  
```cpp  
for (int i = 0; i < weight.size(); i++) { // 正确遍历所有物品 (从0开始)
 for (int j = 0; j <= bagweight; j++) {
 if (j < weight[i]) {
 dp[i][j] = (i == 0) ? 0 : dp[i - 1][j]; // 处理i=0的情况
 } else {
 int prev = (i == 0) ? 0 : dp[i - 1][j - weight[i]];
 dp[i][j] = max((i == 0) ? 0 : dp[i - 1][j], prev + value[i]);
 }
 }
}
```

- \*\*初始化处理\*\*：需显式处理`i=0`的情况（如第一个物品是否可放入背包）。

### \*\*总结\*\*

- \*\*预期功能\*\*：正确实现0-1背包问题的动态规划解法，计算最大价值。

- \*\*代码问题\*\*：外层循环起始值错误导致漏掉第一个物品，需从`i=0`开始遍历所有物品才能得到正确结果。

## 生成式人工智能在软件开发中的应用

# AI辅助编码环境搭建



中南大學  
CENTRAL SOUTH UNIVERSITY

## VSCode + Cline + DeepSeek



Cline v3.4.5

Cline cline.bot | ↗ 779,249 |  
Autonomous coding agent right in yo  
[Disable](#) | [Uninstall](#) |

API提供者

API Key

API Provider: OpenAI Compatible

Base URL: <https://qianfan.baidubce.com/v2>

API Key:

Model ID: deepseek-r1

Set Azure API version

> MODEL CONFIGURATION

(Note: Cline uses complex prompts and works best with Claude models. Less capable models may not work as expected.)

[Let's go!](#)

出现429错误  
要调整TPM

CLINE

Hi, I'm Cline

I can do all kinds of tasks thanks to the latest breakthroughs in [Claude 3.5 Sonnet's agentic coding capabilities](#) and access to tools that let me create & edit files, explore complex projects, use the browser, and execute terminal commands (with your permission, of course). I can even use MCP to create new tools and extend my own capabilities.

To get started, this extension needs an API provider for Claude 3.5 Sonnet.

While Cline currently requires you bring your own API key, we are working on an official accounts system with additional capabilities. Subscribe to our mailing list to get updates!

Enter your email  [Subscribe](#)

API Provider: OpenRouter

OpenRouter API Key:  [Enter API Key...](#)

[Get OpenRouter API Key](#)

This key is stored locally and only used to make API requests from this extension.

[Let's go!](#)

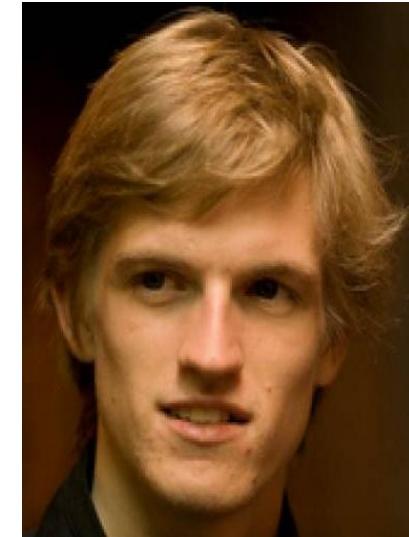
# 框架提高软件开发质量和效率



Flask诞生于2010年，由Armin Ronacher及其团队Pocoo开发。

## Flask框架的特点：

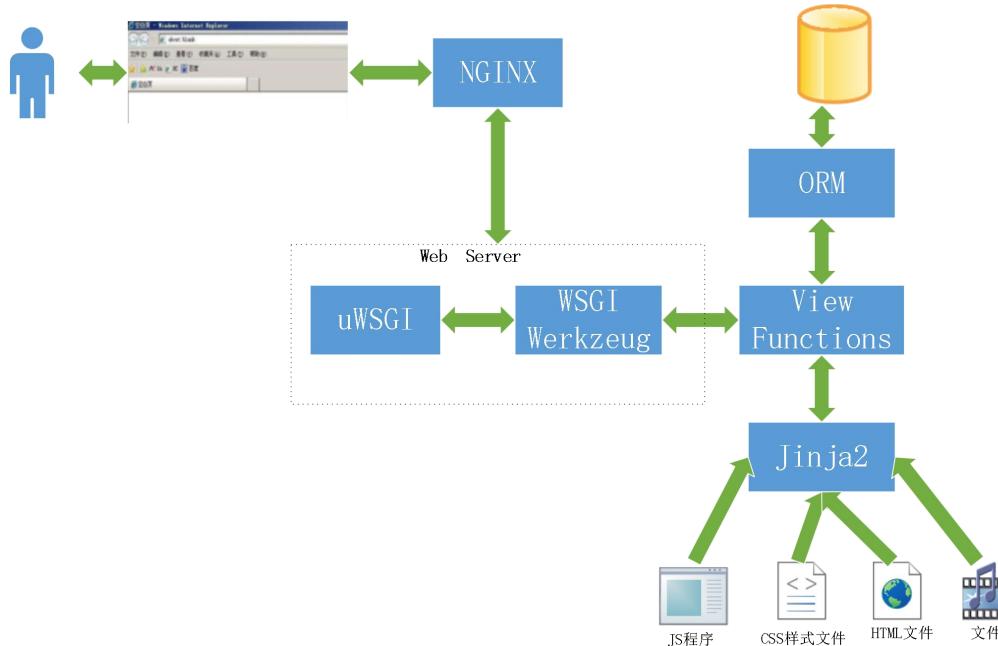
- A small framework (micro-framework)
- An extensible framework
- An MVC framework
- Written in Python
- Two main dependencies:
  - Werkzeug (路由等) (Pocoo开发)
  - Jinja2 (模板等) (Pocoo开发)



框架提高质量和效率



## Flask框架基本架构



框架提高质量和效率



## Flask开发环境搭建

- 安装virtualenv（可选，最好每个项目创建个虚拟环境）
  - sudo apt-get install virtualenv (linux)
  - pip install virtualenv (windows)
  - 构建virtualenv
    - mkdir 项目目录
    - cd 项目目录
    - virtualenv venv
  - 激活环境
    - venv/bin/activate (linux)
    - venv/scripts/common/activate (windows)
- 安装Flask
  - pip install Flask

框架提高质量和效率



## 简单Flask应用

```
from flask import Flask
app = Flask(__name__)
```

WSGI application

tells the application which URL should call the associated function.

```
@app.route('/')
```

URL

```
def hello_world():
 return 'Hello World'
```

view function

```
if __name__ == '__main__':
 app.run()
```

runs the application on the local development server

框架提高质量和效率

# 04 软件开发框架Flask



## 运行结果

```
Console Progress PyUnit
F:\luna\workspace\myFlask\myapp.py
 * Serving Flask app "myapp" (lazy loading)
 * Environment: production
 WARNING: Do not use the development server in a production environment.
 Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```



Hello World

框架提高质量和效率



## Flask应用目录结构

- **app** //程序包
  - **templates** //jinja2模板
    - common //通用模板
    - errors //错误页面
    - user //用户模板
    - email //邮件发送
  - **static** //静态文件
    - js //js脚本文件
    - css //样式表
    - image //图片
    - favicon.ico //网站图标
  - **main** //python程序包，可以有多个这种包，每个对应不同的功能
    - \_\_init\_\_.py
    - errors.py
    - forms //表单处理
    - views //视图处理
    - models //数据库模型
  - **\_\_init\_\_.py**
  - **email.py** //邮件处理程序
  - **config.py** //配置文件
- **migrations** //数据迁移文件夹
- **tests** //单元测试
- **\_\_init\_\_.py**
- **test\*.py** //单元测试程序，可以包含多个对应不同的功能点测试
- **venv** //虚拟环境
- **requirements.txt** //列出了所有依赖包以及版本号，方便在其他位置生成相同的虚拟环境以及依赖
- **config.py** //全局配置文件，配置全局变量
- **manage.py** //启动程序

框架提高质量和效率

# 04 软件开发框架Flask



## Flask应用配置config.py

```
class DevelopmentConfig(Config): #继承Config
 DEBUG = True

 MAIL_SERVER = 'smtp.googlemail.com'
 MAIL_PORT = 587
 MAIL_USE_TLS = True
 MAIL_USERNAME = os.environ.get('MAIL_USERNAME')
 MAIL_PASSWORD = os.environ.get('MAIL_PASSWORD')
 SQLALCHEMY_DATABASE_URI = os.environ.get('DEV_DATABASE_URL') or \
 'sqlite:///+' + os.path.join(basedir, 'data-dev.sqlite')

class TestingConfig(Config):
 TESTING = True
 SQLALCHEMY_DATABASE_URI = os.environ.get('TEST_DATABASE_URL') or \
 'sqlite:///+' + os.path.join(basedir, 'data-test.sqlite')

class ProductionConfig(Config):
 SQLALCHEMY_DATABASE_URI = os.environ.get('DATABASE_URL') or \
 'sqlite:///+' + os.path.join(basedir, 'data.sqlite')

config = [
 'development': DevelopmentConfig,
 'testing': TestingConfig,
 'production': ProductionConfig,
 'default': DevelopmentConfig
]
```

## 使用config.py

```
from flask import Flask
from config import config

app = Flask(__name__)

app.config.from_object(config["development"])
```

框架提高质量和效率

## Flask应用需求文件

- 程序中必须包含一个 `requirements.txt` 文件，用于记录所有依赖包及其精确的版本号。
- 使用如下命令自动生成这个文件：

```
(venv) $ pip freeze >requirements.txt
```

- 如果你要创建这个虚拟环境的完全副本，可以创建一个新的虚拟环境，并在其上运行以下命令：

```
(venv) $ pip install -r requirements.txt
```

框架提高质量和效率



## Flask应用启动脚本manage.py

```
#!/usr/bin/env python
from app import create_app, db
from flask_script import Manager
from flask_migrate import Migrate, MigrateCommand
app = create_app('default')
manager = Manager(app)
migrate = Migrate(app, db)

manager.add_command('db', MigrateCommand)
if __name__ == '__main__':
 manager.run()
```

这个脚本先使用工厂函数创建程序。然后初始化 Flask-Script、Flask-Migrate。出于便利，脚本中加入了 `#!/usr/bin/env python` 声明，所以在基于 Unix 的操作系统中可以通过 `./manage.py` 执行脚本，而不用使用复杂的 `python manage.py`。

框架提高质量和效率

# Flask路由



## Flask动态路由实现机制

- The association between a **URL** and the **function** that handles it is called a **route**.
- Flask类中支持路由功能的数据结构，在`__init__`函数中初始化：

```
url_rule_class = Rule
self.url_map = Map()
self.view_functions = {}
```
- Map和Rule是werkzeug中实现的映射类和路由类。
- Map类先建立了url到endpoint的映射，目的是为了实现动态路由功能。
- view\_functions是一个字典，它负责建立endpoint和视图函数之间的映射关系。

在python安装目录的  
Lib\site-packages\flask目录的app.py文件中定义  
Flask类

框架提高质量和效率



## Flask动态路由实现示例

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
 return 'Hello World'

if __name__ == '__main__':
 print(app.url_map)
 print(app.view_functions)
 #app.run()
```

```
Map([<Rule '/' (HEAD, OPTIONS, GET) ->
hello_world>,
<Rule '/static/<filename>' (HEAD, OPTIONS,
GET) -> static>])

{'hello_world': <function hello_world at
0x013A6570>, 'static': <bound method
Flask.send_static_file of <Flask 'myapp'>>}
```

框架提高质量和效率



## Flask路由传参

- 参数类型: `string`、`int`、`float`
- 如果没有指定参数的类型, 会默认为参数是`string`类型, 给参数指定类型, 就在参数前加上参数类型和冒号即可。

```
@app.route('/<id>/')
def h(id):
 id = int(id) ** 8
 id = str(id)
 return id
```

```
@app.route('/<int:id>/')
def h(id):
 id = id ** 8
 id = str(id)
 return id
```

框架提高质量和效率

## Flask请求方式设置

- flask中请求默认是get请求，若想要指定其他请求方式，用参数**methods**指定。
  - **GET** - 向特定的资源发出请求。
  - **POST** - 向指定资源提交数据进行处理请求（例如提交表单或者上传文件）。
  - **PUT** - 向指定资源位置上传其最新内容。
  - **DELETE** - 请求服务器删除Request-URI所标识的资源。
  - **TRACE** - 回显服务器收到的请求，主要用于测试或诊断。

框架提高质量和效率



## Flask请求方式设置举例：上传文件upload.py

```
coding:utf-8

from flask import Flask,render_template,request,redirect,url_for
from werkzeug.utils import secure_filename
import os

app = Flask(__name__)

@app.route('/upload', methods=['POST', 'GET'])
def upload():
 if request.method == 'POST':
 f = request.files['file']
 basepath = os.path.dirname(__file__) # 当前文件所在路径
 upload_path = os.path.join(basepath, 'static\uploads',secure_filename(f.filename))
 #注意：没有的文件夹一定要先创建，不然会提示没有该路径
 f.save(upload_path)
 return redirect(url_for('upload'))
 return render_template('upload.html')

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

框架提高质量和效率



## Flask请求方式设置举例：上传文件upload.html

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <title>Title</title>
</head>
<body>
 <h1>文件上传示例</h1>
 <form action="" enctype='multipart/form-data' method='POST'>
 <input type="file" name="file">
 <input type="submit" value="上传">
 </form>
</body>
</html>
```

框架提高质量和效率

# Flask模板

## 为何需要模板template

- The obvious task of a view function is to generate a **response** to a request.
- These two types of tasks are formally called **business logic** and **presentation logic**, respectively.
- Mixing business and presentation logic leads to code that is hard to **understand** and **Maintain**.
- Moving the **presentation logic** into **templates** helps improve the maintainability of the application

模板将业务逻辑和表示逻辑解耦合

## 模板引擎template engine

- A template is a file that contains the text of a response, with **placeholder variables** for the dynamic parts that will be known only in the context of a request.
- The process that **replaces the variables with actual values** and returns a final response string is called **rendering** (渲染) .
- For the task of rendering templates, Flask uses a powerful **template engine** called **Jinja2**
- By default, Flask looks for templates in a **templates subdirectory** located inside the main application directory.
- 静态页面模板放在**templates文件**夹中
- 使用**render\_template**函数**渲染模板**
  - 传多个参数的时候可以使用字典

模板将业务逻辑和表示逻辑解耦合

# 04 软件开发框架Flask



## 模板渲染示例：视图函数

```
#coding=utf-8
from flask import Flask,render_template
import time
app = Flask(__name__)
```

注意中文处理：  
`#coding=utf-8`  
中文字符串前加u  
例如： u"王卫东"

```
@app.route('/')
def index():
 class Register(object):
 Email = '123@csu.edu.cn';
 time = str(time.localtime(time.time()).tm_year)+u'年'+ \
 str(time.localtime(time.time()).tm_mon)+u'月' + \
 str(time.localtime(time.time()).tm_mday)+u'日'

 reginf=Register()
 context={
 'username':u"王卫东",
 'age': "22",
 'gender': u"男",
 'register':reginf,
 'wwwurl':{
 'baidu':'www.baidu.com',
 'google':'www.google.com'
 }
 }
 return render_template('index.html', **context)
if __name__ == '__main__':
 app.run(debug=True)
```

单个星号代表这个位置  
接收任意多个非关键字参数，在函数的`*var`位置上将其转化成元组。

而双星号代表这个位置  
接收任意多个关键字参数，在`**var`位置上将其转化成字典。

模板将业务逻辑和表示逻辑解耦合



## 模板渲染示例：模板

The {{ name }} construct used in the template references a variable

```
<!DOCTYPE html>
<html lang="utf-8">
<head>
 <meta charset="UTF-8">
 <title>学习课程申请</title>
</head>
<body>
 <p style="color:#FF00FF">用户名: {{ username }}</p>
 <p>年龄: {{ age }}</p>
 <p style="color:#7B68EE">性别: {{ gender }}</p>
 <!--引用类中的参数-->
 <p style="color:#7B68EE">注册邮箱: {{register.Email}}</p>
 <p style="color:#FF00FF">注册时间: {{register.time }}</p>
 <hr>
 <!--引用字典中的参数,有两种形式-->
 <p style="color:#7B68FF">经常访问的网络平台</p>
 <p style="color:#7B68EE">百度: {{wwwurl['baidu']}}</p>
 <p style="color:#FF00FF">谷歌: {{wwwurl.google|upper}}</p>
</body>
</html>
```

模板将业务逻辑和表示逻辑解耦合



## 模板渲染示例：变量过滤器

Variables can be modified with filters, which are added after the variable name with a pipe character as separator.  
例如：

`{{wwwurl.google|upper}}`

### Jinja2 variable filters

#### Filter name Description

`safe` Renders the value without applying escaping

`capitalize` Converts the first character of the value to uppercase and the rest to lowercase

`lower` Converts the value to lowercase characters

`upper` Converts the value to uppercase characters

`title` Capitalizes each word in the value

`trim` Removes leading and trailing whitespace from the value

`striptags` Removes any HTML tags from the value before rendering

变量过滤器

模板将业务逻辑和表示逻辑解耦合

# 04 软件开发框架Flask



## 模板渲染示例：运行结果



用户名 : 王卫东

年龄 : 22

性别 : 男

注册邮箱 : 123@csu.edu.cn

注册时间 : 2018年12月10日

经常访问的网站平台

百度 : [www.baidu.com](http://www.baidu.com)

谷歌 : [WWW.GOOGLE.COM](http://WWW.GOOGLE.COM)

模板将业务逻辑和表示逻辑解耦合

## 模板控制结构：条件控制

```
{% if user %}
 Hello, {{ user }}!
{% else %}
 Hello, Stranger!
{% endif %}
```

模板将业务逻辑和表示逻辑解耦合



## 模板控制结构：循环控制

```
<dl>
 {% for user in users if not user.hidden %}
 {% if loop.first %}
 <div>User List:</div>
 {% endif %}
 <div class="{{ loop.cycle('odd', 'even') }}">
 <dt>User No. {{ loop.index }}</dt>
 <dd>{{ user.name }}</dd>
 </div>
 {% if loop.last %}
 <dir>Total Users: {{ loop.length }}</dir>
 {% endif %}
 {% else %}
 No users found
 {% endfor %}
 </dl>
```

变量	内容
loop.index	循环迭代计数（从1开始）
loop.index0	循环迭代计数（从0开始）
loop.revindex	循环迭代倒序计数（从len开始，到1结束）
loop.revindex0	循环迭代倒序计数（从len-1开始，到0结束）
loop.first	是否为循环的第一个元素
loop.last	是否为循环的最后一个元素
loop.length	循环序列中元素的个数
loop.cycle	在给定的序列中轮循，如下例在”odd”和”even”两个值间轮循
loop.depth	当前循环在递归中的层级（从1开始）
loop.depth0	当前循环在递归中的层级（从0开始）

Jinja2的循环内置变量

模板将业务逻辑和表示逻辑解耦合

## 模板：宏定义

Jinja2 supports macros, which are similar to functions in Python code.

```
{% macro render_comment(comment) %}
 {{ comment }}
{% endmacro %}
```

```

 {% for comment in comments %}
 {{ render_comment(comment) }}
 {% endfor %}

```

To make macros more reusable, they can be stored in **standalone files** that are then **imported** from all the templates that need them.

```
{% import 'macros.html' as macros %}

 {% for comment in comments %}
 {{ macros.render_comment(comment) }}
 {% endfor %}

```

模板将业务逻辑和表示逻辑解耦合

# 04 软件开发框架Flask



## 模板重用

### 1) 使用include引入

```
{% include 'common.html' %}
```

### 2) 使用模板继承.

```
<html>
 <head>
 {% block head %}
 <title>{% block title %}{% endblock %} - My Application</title>
 {% endblock %}
 </head>
 <body>
 {% block body %}
 {% endblock %}
 </body>
</html>
```

base.html

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

{% block title %}Index{% endblock %}

{% block head %}
{{ super() }}

<style>
</style>

{% endblock %}

{% block body %}
<h1>Hello, World!</h1>

{% endblock %}
```



Hello, World!

模板将业务逻辑和表示逻辑解耦合

## 模板重用举例：Bootstrap集成

Bootstrap 是 Twitter 开发的一个开源框架，它提供的用户界面组件可用于创建整洁且具有吸引力的网页，而且这些网页还能兼容所有桌面和移动端 Web 浏览器。

<https://www.bootstrapcdn.com/>

<https://pypi.org/project/Bootstrap-Flask/#files>

Bootstrap 是客户端框架，不会直接涉及服务器。服务器需要做的只是提供引用了 Bootstrap 层叠样式表（CSS）和 JavaScript 文件的 HTML 响应，以及 HTML、CSS 和 JavaScript 代码中实例化所需组件。这些操作最理想的执行场所就是模版。

**pip install flask-bootstrap**

模板将业务逻辑和表示逻辑解耦合

模板重用举例：Bootstrap集成

## 初始化

Flask extensions are initialized at the same time the application instance is created.

```
from flask_bootstrap import Bootstrap
...
bootstrap = Bootstrap(app)
```

初始化 **Flask-Bootstrap** 之后，就可以在程序中使用一个包含所有**Bootstrap** 文件的基模版。其中就有用来引入 **Bootstrap** 的元素。

模板将业务逻辑和表示逻辑解耦合

# 04 软件开发框架Flask



## 模板重用举例：Bootstrap集成

```
{% extends "bootstrap/base.html" %}

{% block title %}Flasky{% endblock %}

{% block navbar %}
<div class="navbar navbar-inverse" role="navigation">
 <div class="container">
 <div class="navbar-header">
 <button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">
 Toggle navigation

 </button>
 Flasky
 </div>
 <div class="navbar-collapse collapse">
 <ul class="nav navbar-nav">
 Home

 </div>
 </div>
</div>
{% endblock %}
```

## Bootstrap继承

```
{% block content %}
<div class="container">
 <div class="page-header">
 <h1>Hello, {{ username }}!</h1>
 </div>
</div>
{% endblock %}
```

模板将业务逻辑和表示逻辑解耦合

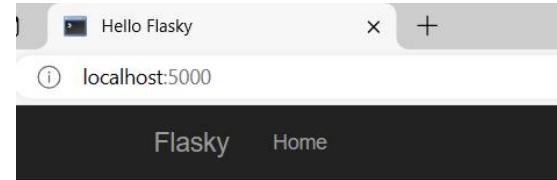
## 模板重用举例：Bootstrap集成

```
#coding=utf-8
from flask import Flask,render_template
import time
app = Flask(__name__)
from flask_bootstrap import Bootstrap
bootstrap = Bootstrap(app)

@app.route('/')
def index():
 return render_template('indexf.html', username=u'王卫东')

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

## 应用及执行结果



Hello, 王卫东!

模板将业务逻辑和表示逻辑解耦合

## Favicon 定义

- A small icon that is used by some browsers to identify a bookmarked Web site.
- The display size of a favicon in the Favorites list is  $16 \times 16$ . Files are recognized by the “.ico” extension.



favicon.ico



```
{% block head %}
{{ super() }}
<link rel="shortcut icon" href="{{ url_for('static', filename='favicon.ico') }}"
type="image/x-icon">
<link rel="icon" href="{{ url_for('static', filename='favicon.ico') }}"
type="image/x-icon">
{% endblock %}
```

模板将业务逻辑和表示逻辑解耦合



# Flask表单

### 什么是表单?

- 前面讨论的模板是单向的，数据只能从服务器到客户端。
- 表单可以通过POST请求将表单数据由web浏览器提交到服务器。服务器端程序通过request.form访问表单数据。
- The Flask-WTF extension makes working with web forms a much more pleasant experience.

**pip install flask-wtf**

表单用来将数据从客户端传送到服务器

## 避免跨站攻击的配置

- 在应用根目录下创建一个配置文件config.py,其内容如下:

```
CSRF_ENABLED = True
```

```
SECRET_KEY = 'hard-to-guess'
```

- 在应用根目录下创建一个应用myserver.py文件，其内容如下:

```
from flask import Flask
```

```
app = Flask(__name__)
```

```
app.config.from_object('config')
```

表单用来将数据从客户端传送到服务器



## 表单类

- 使用**Flask-WTF**创建的表单就像一个对象，需要从**Form**类继承子类。然后在这个子类中定义一些类的属性变量作为表单字段就可以了。
- 在应用根目录下创建一个**forms.py**文件

```
config=utf-8
from flask_wtf import FlaskForm as Form
from wtforms import StringField, PasswordField
from wtforms.validators import DataRequired
class LoginForm(Form):
 accountNumber = StringField('accountNumber', validators=[DataRequired('accountNumber is null')])
 password = PasswordField('password', validators=[DataRequired('password is null')])
```

表单用来将数据从客户端传送到服务器

# 04 软件开发框架Flask



## WTForms 支持的HTML 字段

Field type	Description
BooleanField	Checkbox with True and False values
DateField	Text field that accepts a <code>datetime.date</code> value in a given format
DateTimeField	Text field that accepts a <code>datetime.datetime</code> value in a given format
DecimalField	Text field that accepts a <code>decimal.Decimal</code> value
FileField	File upload field
HiddenField	Hidden text field
MultipleFileField	Multiple file upload field
FieldList	List of fields of a given type
FloatField	Text field that accepts a floating-point value
FormField	Form embedded as a field in a container form
IntegerField	Text field that accepts an integer value
PasswordField	Password text field
RadioField	List of radio buttons
SelectField	Drop-down list of choices
SelectMultipleField	Drop-down list of choices with multiple selection
SubmitField	Form submission button
StringField	Text field
TextAreaField	Multiple-line text field

## WTForms 验证函数

Validator	Description
DataRequired	Validates that the field contains data after type conversion
Email	Validates an email address
EqualTo	Compares the values of two fields; useful when requesting a password to be entered twice for confirmation
InputRequired	Validates that the field contains data before type conversion
IPAddress	Validates an IPv4 network address
Length	Validates the length of the string entered
MacAddress	Validates a MAC address
NumberRange	Validates that the value entered is within a numeric range
Optional	Allows empty input in the field, skipping additional validators
Regexp	Validates the input against a regular expression
URL	Validates a URL
UUID	Validates a UUID
AnyOf	Validates that the input is one of a list of possible values
NoneOf	Validates that the input is none of a list of possible values

表单用来将数据从客户端传送到服务器



## 表单示例

- 显示登录表单的模板login.html

```
<!-- extend from base layout -->
{% extends "bootstrap/base.html" %}

{% block content %}
<h1>Sign In</h1>
<form action="" method="post" name="login">
 {{form.hidden_tag()}}
 <p>
 Account number:

 {{form.accountNumber(size=80)}}

 </p>
 <p>Password:

 {{form.password}}

 </p>
 <p><input type="submit" value="Sign In"></p>
</form>
{% endblock %}
```

```
from flask_bootstrap import Bootstrap
bootstrap = Bootstrap(app)
```

Bootstrap集成

表单用来将数据从客户端传送到服务器



## 表单示例

- 表单视图： MyServer.py程序

```
from flask import Flask
app = Flask(__name__)
app.config.from_object('config')

from flask_bootstrap import Bootstrap
bootstrap = Bootstrap(app)

from flask import render_template
from forms import LoginForm

@app.route('/login', methods = ['GET', 'POST'])
def login():
 form = LoginForm()
 return render_template('login.html', form = form)

if __name__ == '__main__':
 app.run()
```

## 运行结果

The screenshot shows a web browser window with the URL `http://127.0.0.1:5000/login` in the address bar. The page title is "Sign In". There are two input fields: one for "Account number" containing "sss" and another for "Password" containing "\*\*\*\*\*". Below the password field is a "Sign In" button.

表单用来将数据从客户端传送到服务器



# Web应用数据库如何连接？

## Flask-SQLAlchemy

Flask本身没有内置ORM框架，需要依赖第三方模块。Flask-SQLAlchemy 是 Flask 的扩展，本质上是对SQLAlchemy的进一步封装。

SQLAlchemy是一个基于Python实现的ORM框架：将类和对象转换成SQL，然后使用数据API执行SQL并获取执行结果。

## ORM的优势

- 大大缩短了程序员的编码时间，减少甚至免除了对Model的编码
- 良好的数据库操作接口，使编码难度降低，使团队成员的代码变得简洁易读、风格统一
- 动态的数据表映射，在数据表结构甚至数据库发生改变时，减少了相应的代码修改
- 减少了程序员对数据库的学习成本
- 可以很方便地引入数据缓存之类的附加功能



## 数据库配置

`pip install flask-sqlalchemy`

In Flask-SQLAlchemy, a database is specified as a URL.

Database engine	URL
MySQL	<code>mysql://username:password@hostname/database</code>
Postgres	<code>postgresql://username:password@hostname/database</code>
SQLite (Linux, macOS)	<code>sqlite:///absolute/path/to/database</code>
SQLite (Windows)	<code>sqlite:///c:/absolute/path/to/database</code>

程序使用的数据库 URL 必须保存到 Flask 配置对象的 `SQLALCHEMY_DATABASE_URI` 键中。

```
app.config['SQLALCHEMY_DATABASE_URI'] =
'mysql+pymysql://root:root@localhost:3306/student'
```



## 使用MySQL数据库

安装驱动: `pip install pymysql`

配置config.py的内容

`CSRF_ENABLED = True`

`SECRET_KEY = 'you-will-never-guess'`

`SQLALCHEMY_DATABASE_URI =`

`'mysql+pymysql://root:123456@localhost:3306/student'`

`SQLALCHEMY_TRACK_MODIFICATIONS = False`

`SQLALCHEMY_COMMIT_ON_TEARDOWN = True`

`SQLALCHEMY_COMMIT_ON_TEARDOWN` 键,将其设为 `True`时,每次请求结束后都会自动提交数据库中的变动。

`SQLALCHEMY_TRACK_MODIFICATIONS` 键,将其设为 `False`时,可以减少内存使用。

## MySQL数据库操作：进入mysql

- 进入MySQL: mysql -h <主机地址> -u <用户名> -p <密码>

```
C:\Users\rsb>mysql -uroot -p
Enter password: ****
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 6
Server version: 5.5.40 MySQL Community Server (GPL)

Copyright (c) 2000, 2014, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> |
```

- root用户初始密码: root (安装时给定)



## MySQL数据库操作：创建用户

- 采用insert创建用户：

在my.ini配置文件指定了严格模式，禁止通过insert直接修改mysql库中的user表进行添加新用户

```
mysql> insert into mysql.user(Host, User, Password) values('localhost','dml',password('123456'));
ERROR 1364 (HY000): Field 'ssl_cipher' doesn't have a default value
```

可以修改my.ini，去掉安全模式，即去掉sql-mode中的STRICT\_TRANS\_TABLES，重启mysql或者：

```
mysql> insert into mysql.user(Host, User, Password, ssl_cipher, x509_issuer, x509_subject)
values('localhost','zlh',password('123456'),","","","");
Query OK, 1 row affected (0.00 sec)
```

- 采用grant创建用户

```
grant all on mysql.user to maowei@localhost identified by "123456";
```

- 采用create创建用户

```
create user 'john'@localhost identified by "123";
```

- 刷新系统权限表: flush privileges;

```
mysql> select host,user,password from mysql.user;
+-----+-----+-----+
| host | user | password |
+-----+-----+-----+
| localhost | root | *81F5E21E35407D884A6CD4A731AEFB6AF209E1B |
| localhost | zlh | *6BB4837EB74329105EE4568DDA7DC67ED2CA2AD9 |
| % | root | *6BB4837EB74329105EE4568DDA7DC67ED2CA2AD9 |
| localhost | wkp | *6BB4837EB74329105EE4568DDA7DC67ED2CA2AD9 |
| localhost | maowei | *6BB4837EB74329105EE4568DDA7DC67ED2CA2AD9 |
| localhost | john | *23AE809DDACAF96AF0FD78ED04B6A265E05AA257 |
+-----+-----+-----+
6 rows in set (0.00 sec)
```



## MySQL数据库操作: 创建数据库

- 创建数据库: `create database <数据库名>;`  
`create database libman_db;`
- 显示创建的数据库: `show databases;`

```
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| libman_db |
| mysql |
| performance_schema |
| test |
+-----+
5 rows in set (0.00 sec)
```

- 删 除 数据库 `drop database <数据库名>;`

```
mysql> drop database libman_db;
Query OK, 0 rows affected (0.00 sec)

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| test |
+-----+
4 rows in set (0.00 sec)
```

## 退出mysql

```
mysql> quit;
Bye

C:\Users\rsb>|
```

## MySQL数据库操作: 创建数据库表

- 打开数据库: `use <数据库名>;`  
`use libman_db;`
- 显示数据库中的表: `show tables;`
- 创建数据库表

```
CREATE TABLE students (
 id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
 name VARCHAR(30) NOT NULL,
 birthday DATE NOT NULL
);
```

- 显示库结构: `desc <表名>;`

```
mysql> desc students;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id | int(6) unsigned | NO | PRI | NULL | auto_increment |
| name | varchar(30) | NO | | NULL | |
| birthday | date | NO | | NULL | |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```



## SQLAlchemy数据库操作: 定义库表模型

```
from flask import Flask
app = Flask(__name__)
app.config.from_object('config')

from flask_sqlalchemy import SQLAlchemy
Create database connection object
db = SQLAlchemy(app)

class Role(db.Model):
 __tablename__ = 'roles'
 id = db.Column(db.Integer, primary_key=True) #定义主键
 name = db.Column(db.String(64), unique=True)
 users = db.relationship('User', backref='role') #定义关系

class User(db.Model):
 __tablename__ = 'users'
 id = db.Column(db.Integer, primary_key=True)
 username = db.Column(db.String(64), unique=True, index=True)
 email = db.Column(db.String(128), unique=False, index=True)
 role_id = db.Column(db.Integer, db.ForeignKey('roles.id')) #定义外键
```

mydatabase.py 的内容



## SQLAlchemy数据库操作: 定义库表模型执行结果

config.py的内容如下:

```
CSRF_ENABLED = True
SECRET_KEY = '771177273854'
SQLALCHEMY_DATABASE_URI =
'mysql+pymysql://root:root@localhost:3306/libman_db'
SQLALCHEMY_TRACK_MODIFICATIONS = False
SQLALCHEMY_COMMIT_ON_TEARDOWN = True
```

- 首先在mysql中创建数据库libman\_db;
- 然后运行createtable.py创建数据库表:  
`python createtable.py`

```
mysql> use libman_db;
Database changed
mysql> show tables;
+-----+
| Tables_in_libman_db |
+-----+
| roles
| students
| users
+-----+
3 rows in set (0.00 sec)
```

```
mysql> desc roles;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra
+-----+-----+-----+-----+-----+
| id | int(11) | NO | PRI | NULL | auto_increment
| name | varchar(64)| YES | UNI | NULL |
+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

```
mysql> select * from users;
+-----+-----+-----+-----+
| id | username | email | role_id |
+-----+-----+-----+-----+
| 1 | admin | admin@example.com | NULL |
| 2 | guest | guest@example.com | NULL |
+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

createtable.py的内容如下:

```
from mydatabase import db, app, User
import pymysql

if __name__ == '__main__':
 with app.app_context(): #新版本需要特别声明上下文
 db.drop_all()
 db.create_all()
 admin = User(username='admin', email='admin@example.com')
 guest = User(username='guest', email='guest@example.com')
 db.session.add(admin)
 db.session.add(guest)
 db.session.commit()
```

```
mysql> desc users;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra
+-----+-----+-----+-----+-----+
| id | int(11) | NO | PRI | NULL | auto_increment
| username | varchar(64) | YES | UNI | NULL |
| email | varchar(128) | YES | MUL | NULL |
| role_id | int(11) | YES | MUL | NULL |
+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

## SQLAlchemy数据库操作:库表查询

➤ 数据查询: data = User.query.all()

➤ 过滤查询:

```
data = User.query.filter(User.username=='wangli',User.age==15)
for u in data:
 print(u.username,u.age)
```

➤ 单条件查询

```
data = User.query.filter_by(age=18)
```

➤ 模糊查询

```
data = User.query.filter(User.username.like('%wang%')) #包含wang
data = User.query.filter(User.username.like('%wang')) #以wang作为结尾
data = User.query.filter(User.username.like('wang%')) #以wang作为开头
```

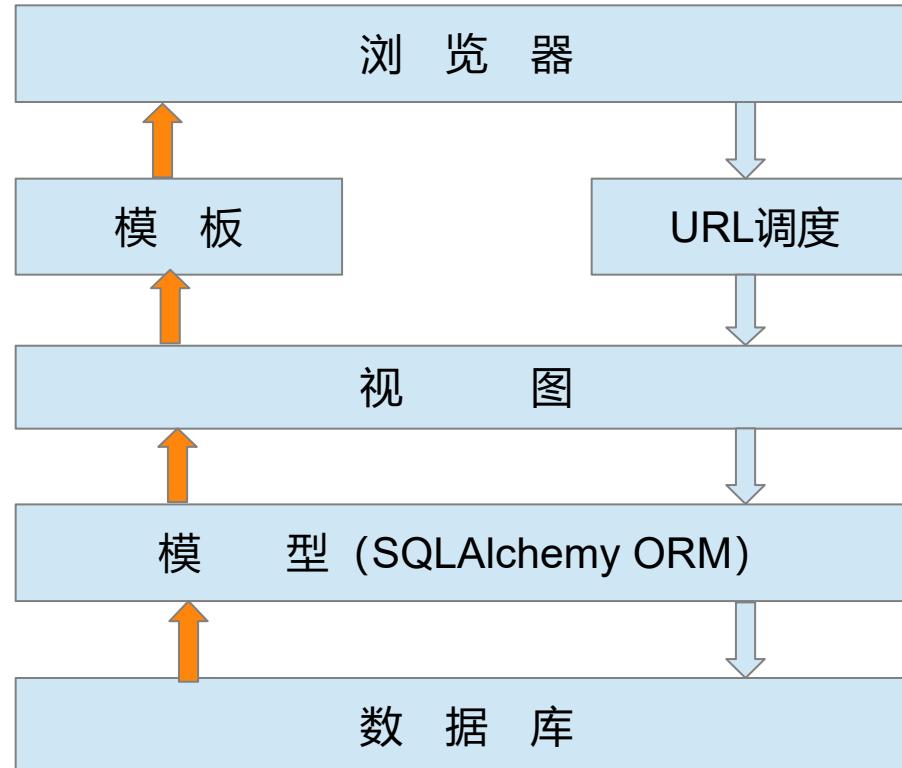
## SQLAlchemy数据库操作: 数据库逻辑查询

```
from sqlalchemy import and_, or_, not_
data = User.query.filter(User.sex==True,User.age<15)
data = User.query.filter(User.sex==True).filter(User.age<15)
data = User.query.filter(and_(User.sex==True,User.age<15))
data = User.query.filter(not_(User.age<=15))
```

# 基于Flask的web应用特点是什么？



## 示例应用技术架构：MTV Architecture





## 示例应用：基本目录结构

```
▽ STUDENTMICROBLOG
 ▽ studentapp
 > static
 > templates
 ⚡ config.py
 ⚡ model.py
 ⚡ mystudentapp.py
 ⚡ server.py
 ⚡ studentForm.py
 ⚡ view.py
```

- 项目目录: StudentMicroblog
- 应用目录: studentapp
- 静态文件目录: static
- 模板目录: templates
  
- 配置程序: config.py
- 库表程序: model.py
- 应用初始化程序: mystudentapp.py
- 应用服务程序: server.py
- 学生表单程序: studentForm.py
- 视图函数程序: view.py

## SQLAlchemy数据库操作: 定义ORM

```
from mystudentapp import db
```

```
class User(db.Model):
```

```
 __tablename__ = 'users'
```

```
 id = db.Column(db.Integer, primary_key=True)
```

```
 nickname = db.Column(db.String(64), unique=True)
```

```
 pwd = db.Column(db.String(11), nullable=True)
```

```
 posts = db.relationship('Post', backref='user')
```

#设置表名

#主键

#字段nickname

#字段pwd

#与Post是一对多

#一对一：把userlist参数设置为False

```
class Post(db.Model):
```

```
 __tablename__ = 'posts'
```

```
 id = db.Column(db.Integer, primary_key=True)
```

```
 author = db.Column(db.String(64), nullable=True)
```

```
 content = db.Column(db.String(256), nullable=True)
```

```
 user_id = db.Column(db.Integer, db.ForeignKey('users.id')) #外键
```

## model.py的内容

## 04 Flask框架应用示例：学生微博



### SQLAlchemy数据库操作: 编写应用

#### mystudentapp.py 的内容

```
from flask import Flask
import config

app = Flask(__name__) # 声明应用

app.config.from_object(config) # 加载config.py配置文件
app.config['SECRET_KEY'] = '03e6f8g4' # 一般借助os模块用环境变量

from flask_sqlalchemy import SQLAlchemy

db = SQLAlchemy(app) # 声明数据库

import view # 必须有，否则找不到视图函数
```

#### config.py 的内容

#### server.py 的内容

```
from mystudentapp import app

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

```
SQLALCHEMY_DATABASE_URI =
'mysql+pymysql://root:root@localhost/students'
SQLALCHEMY_TRACK_MODIFICATIONS = True
SQLALCHEMY_COMMIT_ON_TEARDOWN = True
```

## SQLAlchemy数据库操作:利用flask shell操作ORM

- 进入应用ORM文件model.py所在的目录，启动终端。
- 设置环境变量FLASK\_ENV
  - windows下: set FLASK\_ENV=development
  - linux下: export FLASK\_ENV=development

- 启动flask shell: flask --app server.py shell

```
>>> from mystudentapp import db
>>> from model import User, Post
>>> db.drop_all()
>>> db.create_all()
```

```
mysql> show tables;
+-----+
| Tables_in_students |
+-----+
| posts
| users
+-----+
```

```
mysql> desc users;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra
+-----+-----+-----+-----+-----+-----+
| id | int(11) | NO | PRI | NULL | auto_increment
| nickname | varchar(64) | YES | UNI | NULL |
| pwd | varchar(11) | YES | | NULL |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

```
mysql> desc posts;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra
+-----+-----+-----+-----+-----+-----+
| id | int(11) | NO | PRI | NULL | auto_increment
| author | varchar(64) | YES | | NULL |
| content | varchar(256) | YES | | NULL |
| user_id | int(11) | YES | MUL | NULL |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

## SQLAlchemy数据库操作：利用flask shell 操作ORM

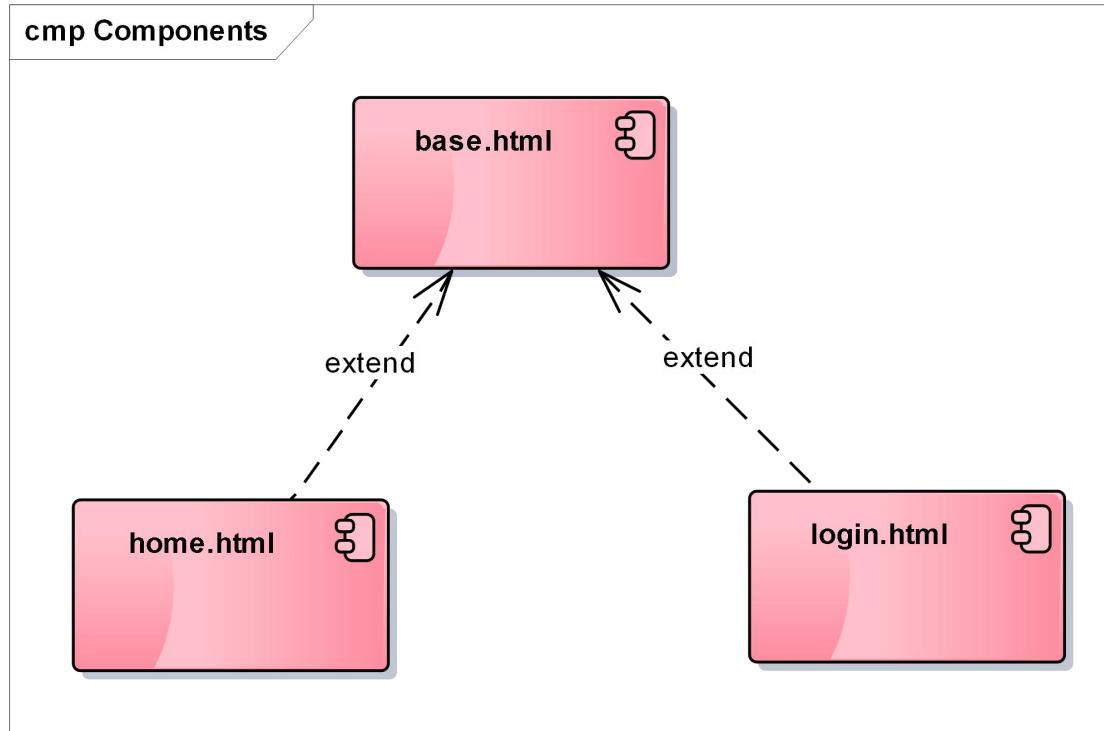
➤ 向库表添加记录

```
>>> us1= User(nickname='liming', pwd='123456')
>>> ps1 = Post(author='renhui', content='This movie is very cool!', user=us1)
>>> us2= User(nickname='renhui', pwd='123456')
>>> ps2 = Post(author='renhui', content='This movie is very cool!', user=us2)
>>> db.session.add_all([us1,us2,ps1,ps2])
>>> db.session.commit()
```

```
mysql> select * from users;
+----+-----+-----+
| id | nickname | pwd |
+----+-----+-----+
| 1 | liming | 123456 |
| 2 | renhui | 123456 |
+----+-----+-----+
2 rows in set (0.00 sec)
```

```
mysql> select * from posts;
+----+-----+-----+-----+
| id | author | content | user_id |
+----+-----+-----+-----+
| 1 | renhui | This movie is very cool! | 1 |
| 2 | renhui | This movie is very cool! | 2 |
+----+-----+-----+-----+
2 rows in set (0.00 sec)
```

## 应用模板





## 应用模板

# base.html的内容

```
<html>
 <head>
 {% if title %}
 <title>{{title}} - microBlog</title>
 {% else %}
 <title>microBlog</title>
 {% endif %}
 </head>
 <body>
 <div>MicroBlog:
 Home
 Login
 </div>
 <hr>
 <hr>
```

```
{% with messages = get_flashed_messages() %}
{% if messages %}

 {% for message in messages %}
 {{message}}
 {% endfor %}

{% endif %}
{% endwith %}
{% block content %}{% endblock %}
</body>
</html>
```

## 应用模板

### home.html的内容

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

{% block content %}
 <h1>Hello, {{user.nickname}}!</h1>
 {% for post in posts %}
 <p>{{post.author.nickname}} says:
{{post.body}}
</p>
 {% endfor %}
{% endblock %}
```

## 应用模板

# login.html的内容

```
{% extends "base.html" %}
{% block content %}
 <h1>Sign In</h1>
 <form action="" method="post">
 {{ form.hidden_tag() }}
 <p>
 {{ form.username.label }}

 {{ form.username(size=32) }}

 {% for error in form.username.errors %}
 [{{ error }}]
 {% endfor %}
 </p>
 <p>
 {{ form.password.label }}

 {{ form.password(size=32) }}

 {% for error in form.password.errors %}
 [{{ error }}]
 {% endfor %}
 </p>
 <p>{{ form.remember_me() }} {{ form.remember_me.label }}</p>
 <p>{{ form.submit() }}</p>
 </form>
{% endblock %}
```

## 应用表单

# studentForm.py的内容

```
{% extends "base.html" %}
{% block content %}
 <h1>Sign In</h1>
 <form action="" method="post">
 {{ form.hidden_tag() }}
 <p>
 {{ form.username.label }}

 {{ form.username(size=32) }}

 {% for error in form.username.errors %}
 [{{ error }}]
 {% endfor %}
 </p>
 <p>
 {{ form.password.label }}

 {{ form.password(size=32) }}

 {% for error in form.password.errors %}
 [{{ error }}]
 {% endfor %}
 </p>
 <p>{{ form.remember_me() }} {{ form.remember_me.label }}</p>
 <p>{{ form.submit() }}</p>
 </form>
{% endblock %}
```



## 应用视图

# view.py的内容

```
from mystudentapp import app

from flask import render_template
from flask import flash, redirect
from studentForm import LoginForm
from model import User, Post

@app.route('/home',methods=['get'])
@app.route('/',methods=['get'])
def home():
 user = {'nickname': 'Bobm'}
 posts = [
 {
 'author': { 'nickname': 'John' },
 'body': 'Beautiful day in Portland!'
 },
 {
 'author': { 'nickname': 'Susan' },
 'body': 'The Avengers movie was so cool!'
 }
]
 return render_template('home.html',
 title = 'home',
 user = user,
 posts = posts)
```

```
@app.route('/login',methods=['get', 'post'])
def login():
 form = LoginForm()
 if form.validate_on_submit():
 flash('Login requested for user={}'.format(form.username.data))
 remember_me='{}'.format(form.username.data, form.remember_me.data)
 user = User.query.filter_by(nickname=form.username.data).first()
 if user is not None and user.pwd==form.password.data:
 post = Post.query.filter_by(user_id = user.id).first()
 posts=[{
 'author': { 'nickname': post.author },
 'body': post.content
 }]
 else:
 user = {}
 posts = []
 return render_template('home.html',
 title = 'home',
 user = user,
 posts = posts)

 return render_template('login.html',
 title='Sign In', form=form)
```

# 04 Flask框架应用示例：学生微博



## 启动应用server.py

```
PS D:\pythonProjects\StudentMicroBlog\studentapp> python server.py
 * Serving Flask app 'mystudentapp'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 691-354-415
```

This screenshot shows a browser window with the title 'home - microBlog'. The address bar contains '127.0.0.1:5000'. The page content includes a 'MicroBlog' header with 'Home Login' links, and a main section displaying a message from 'Bobm': 'Hello, Bobm!'. Below this, there are two entries: 'John says: Beautiful day in Portland!' and 'Susan says: The Avengers movie was so cool!'. The 'Login' button in the top right corner is highlighted with a red circle.

This screenshot shows a browser window with the title 'Sign In - microBlog'. The address bar contains '127.0.0.1:5000/login'. The page content includes a 'MicroBlog' header with 'Home Login' links, and a main section titled 'Sign In' with fields for 'User Name' (containing 'liming') and 'Password' (containing '\*\*\*\*\*'). There is also a 'Remember me' checkbox and a 'Sign In' button at the bottom.

This screenshot shows a browser window with the title 'home - microBlog'. The address bar contains '127.0.0.1:5000/login'. The page content includes a 'MicroBlog' header with 'Home Login' links, and a main section displaying a message from 'liming': 'Hello, liming!'. A log entry at the bottom left states: '• Login requested for user=liming, remember\_me=True'. The 'Login' button in the top right corner is highlighted with a red circle.

Hello, Bobm!

John says:  
Beautiful day in Portland!

Susan says:  
The Avengers movie was so cool!

Sign In

User Name  
liming

Password  
\*\*\*\*\*

Remember me

Sign In

Hello, liming!

renhui says:  
This movie is very cool!



中南大學  
CENTRAL SOUTH UNIVERSITY

感谢各位聆听！  
祝大家学习愉快！