Flask,sqlalchemy实用文档：

sqlanlchemy文档：

**插入一列**

**def** add\_column(engine, table\_name, column):  
 column\_name = column.compile(dialect=engine.dialect)  
 column\_type = column.type.compile(engine.dialect)  
 engine.execute(**'ALTER TABLE %s ADD COLUMN %s %s'** % (table\_name, column\_name, column\_type))  
  
column = Column(**'topic'**, String(100))  
add\_column(engine, **'essay'**, column)

Flask文档：

-1、login\_manager放的位置：

app = Flask(\_\_name\_\_)  
app.config[**'SECRET\_KEY'**] = **'123456'**login\_manager = LoginManager()  
login\_manager.login\_view = **'sign.sign\_in'**login\_manager.init\_app(app)  
@login\_manager.user\_loader  
**def** load\_user(user\_id):  
 **return** Users.query.get(int(user\_id))

-2、login\_user(result, remember=**True**)的result是一个user对象，一定要是查的到的才行

-1.注意form表单提交的时候页面就变成了self的覆盖当前页面。

1. 所有的表单验证应该在前端验证好。

关于form的表单登录或注册问题；一般情况下是在前端做验证；之后就跳转在后台做验证；

后台验证失败在重新把之前的内容插入html中；返回html并告知错误的地方。

需求一。实际写代码的时候要搞定表单的后台通过以后的回调函数；而不是直接的跳转；回调成功以后跳转；否则不跳转。

1. Flask之blueprint的路由访问；

第一次的view-----app.register\_blueprint(article, url\_prefix=**'/article'**)

就已经定义了路由访问的第一个路径；后面在article这个py文件里面再次定义就是定义第二个路径

article=Blueprint(**'article'**,\_\_name\_\_)  
@article.route(**'/article'**)  
**def** index():  
 **return** render\_template(**'article.html'**)

则需要这样才能访问到http://127.0.0.1:5000/article/article

1.01 在蓝图里面的视图函数不可以有重复的名称

1.02如果路由是http://127.0.0.1:5000/catlog

当前域名下的节点：

那么当**window**.open(**'./sign/sign\_in'**,**'\_self'**); *//覆盖当前页面打开*

实际请求的路由就是http://127.0.0.1:5000/catlog/sign/sign\_in

解决方法 ：在根目录下的路由请求：

当期域名：

**window**.open(**'../sign/sign\_up'**,**'\_self'**); *//覆盖当前页面打开*

1. 引入静态文件：

<script src="/static/js/ion.calendar.min.js"></script>

1. static访问根路径



1. flask-sqlalchemy 关于使用sqllite更快的数据查询

app.config[**'SQLALCHEMY\_DATABASE\_URI'**]=**'sqlite:///./sqljinmi.db'**app.config[**'SQLALCHEMY\_TRACK\_MODIFICATIONS'**]=**False #不需要跟踪对象变化时降低内存消耗**

db=SQLAlchemy(app)  
**class** JinMi(db.Model):  
 \_\_tablename\_\_=**'essay'** id = db.Column(db.Integer,primary\_key=**True**)  
 title=db.Column(db.String(50))  
 one\_label=db.Column(db.String(30))  
 two\_label=db.Column(db.String(30))  
 author=db.Column(db.String(20))  
 visiting=db.Column(db.String(20))  
 discuss=db.Column(db.String(20))  
 like=db.Column(db.String(20))  
 tupian=db.Column(db.String(100))  
 content=db.Column(db.String)  
 publish\_date=db.Column(db.String(20))  
 unique\_pid=db.Column(db.String(40))

x=JinMi.query.order\_by(**'-like'**).limit(5).all()

按照like 从大到小限定5个排序

1. **启用safe过滤器 避免html标签转义**

**var *others***={{other|safe}};

1. **在script里面使用变量；如果变量是html标签则变成字符串类型：**

other=**'"<p>nihao</p>"'**

**7.for 循环段落然后放入<p>中先replace "\r"**

**7.获取form表单数据**

@sign.route(**'/register'**,methods=[**'POST'**])  
**def** register():  
 print(dict(request.form))

8.username = db.Column(db.String(20),unique=**True**) *#设置为不可重复*

1. flask\_sqlalchemy创建数据表

db.create\_all() 只要这句即可，不会报错，他会判断创建的就不创建了，没创建的才创建。

1. 判断是否注册：

result = Users.query.filter\_by(nickname=nickname).first()

1. 重定向跳转

**return** redirect(**'../catlog'**)

1. 路由添加参数：

**return** redirect(**'../catlog?nickname={}'**.format(nickname))

1. 获取get请求的参数

print(dict(request.args))

1. 删除jinja2 的双引号

***nickname*** = ***nickname***.replace(**"\""**, **""**);