侯赛雷

昵称： 侯赛雷
园龄： 1年10个月
粉丝： 32
关注： 4
+加关注

< 2020年2月 >

日	一	二	三	四	五	六
26	27	28	29	30	31	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
1	2	3	4	5	6	7

flask插件全家桶集成学习---持续更新ing

目录

正文

不得不说flask的设计要比django要小巧精妙的多了,没有那么臃肿,只保留核心功能,其他的都需要自己引入,即各种各样的插件来满足我们的需求,我这里记录一下自己学习项目中用的插件使用方法和一些技巧总结!

常用链接

我的随笔

我的评论

我的参与

最新评论

我的标签

随笔分类

Docker(2)

groovy(6)

guava(2)

hadoop(1)

java(38)

JDK源码(16)

jvm

linux(5)

Lua(1)

mongodb

先放一下代码地址：https://gitee.com/houzheng1216/pythonxuexi/tree/master/flask/fristflask

回到顶部

— flask-cli

falsk内置的脚手架,可代替flask-script管理项目,再也不用写manager.py了

启动项目：根目录命令：flask run

开启shell： flask shell

开启一个交互式的python shell，用来访问或处理应用数据。该指令默认激活应用上下文，并导入应用实例。
只有应用实例是默认导入的，如果需要导入其他对象，使用shell_context_processor装饰函数，返回一个字典对象，键值对表示额外导入的对象。

导入：

```
# 在flask shell 中导入其他对象,导入后shell可以使用
@app.shell_context_processor
def make_shell_context():
    return dict(models=models, db=db)
```

开启之后默认可查看app:

自定义命令：

```
# 自定义命令,分组命令,使用user管理多个命令

user_cli = AppGroup("user")


@user_cli.command("print")  # 指定命令
```

```
@click.argument("name")  # 指定参数
def print_user(name):
    print("this is", name)


@user_cli.command("add")
@click.argument("num")
def add_num(num):
    print("result is", num+num)
```

app中注册命令:

```
# 添加自定义命令
app.cli.add_command(user_cli)
```

使用命令,flask 分组名称 命令名称 参数:

回到顶部

二 flask-debugToolbar

这个调试贼好用,简单安装配置使用就完事了

```
# 配置debugToolbar
toolbar = DebugToolbarExtension()
app.config['SECRET_KEY'] = 'hou'
app.config['DEBUG_TB_INTERCEPT_REDIRECTS'] = False  # 不拦截重定向
toolbar.init_app(app)
```

使用,页面右边:

这里贴一些常用config配置:

前端(11)
设计模式(4)
数据结构
搜索引擎(4)
算法(1)
微信小程序(3)

随笔档案
2020年2月(6)
2020年1月(12)
2019年12月(8)
2019年11月(15)
2019年10月(6)
2019年9月(1)
2019年8月(6)
2019年7月(7)
2019年6月(7)
2019年5月(13)

回到顶部

三 flask-WTF

回到顶部

四 falsk-login

Flask-Login 为 Flask 提供用户 session 的管理机制。它可以处理 Login、Logout 和 session 等服务。
将用户的 id 储存在 session 中，方便用于 Login/Logout 等流程。
让你能够约束用户 Login/Logout 的视图
提供 remember me 功能
保护 cookies 不被篡改,这个可以和flask-admin一起使用,很方便,适合简单轻量级的需求

不过新出了更强大的权限框架Pycasbin

定义User模型:

```
''' flask-login user类必须实现以下:
        is_authenticated 是否属性
        is_active 是否激活属性
        is_anonymous 是否匿名属性
        get_id() 方法, 可以继承UserMixin,提供了默认实现
'''
class User(db.Model, UserMixin):
    __tablename__ = 'user'  # 表名
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(50))
    password = db.Column(db.String(50))
    age = db.Column(db.Integer)
```

2019年4月(10)
2019年3月(4)
2019年1月(9)
2018年12月(6)
2018年11月(8)
2018年10月(17)
2018年9月(9)
2018年8月(8)
2018年7月(1)
2018年6月(5)
2018年5月(7)
2018年4月(12)

文章分类

Vertx

相册

bg(2)

```
    write_time = db.Column(db.DateTime, default=datetime.now())
    # toString方法
    def __repr__(self):
        return '<User %r>' % self.name
```

配置:

```
# 配置flask-login
login_manager = LoginManager()
login_manager.login_view = 'myuser.login' # 指定跳转登录函数,权限拦截时跳转
login_manager.login_message_category = 'info'
login_manager.login_message = '请先登录'
login_manager.init_app(app)

# 配置回调函数,必须配置
# user session 记录的是用户 ID (user_id), 回调函数的作用就是通过 user_id 返回对应的 User 对象
@login_manager.user_loader
def load_user(user_id):
    print(user_id)  # 登录成功的时候会自动获取主键id
    return models.User.query.filter_by(id=user_id).first()
```

登录登出方法:

```
@user.route('/login/', methods=['GET', 'POST'])
def login():
    form = LoginForm()  # 实例化form对象
    if request.method == "POST" and form.validate_on_submit():
        username = request.form.get('username')
```

```
        password = request.form.get('password')
        user = User.query.filter_by(name=username, password=password).first()
        if user:
            # 将用户信息注册到flask-login中
            login_user(user)
            return redirect(url_for('myuser.get_all_user'))
        else:
            flash("请重新输入用户名和密码")
    return render_template('login.html', form=form)

@user.route('/logout/', methods=['GET', 'POST'])
@login_required   # 装饰器,必须登录才能有权限
def logout():
    logout_user()
    return redirect(url_for('myuser.get_all_user'))
```

权限控制:

使用@login_required 注解拦截函数

和admin一起使用,拦截后台数据管理页面,不会admin的可以先看flask-admin:

```
class MyModelView(ModelView):
    can_delete = True
    can_create = False  # 是否能创建
    # Override displayed fields
    column_list = ('name', 'age')  # 显示的属性
    # 重写方法实现权限控制
    def is_accessible(self):
        print(current_user.name)  #  获取login的当前用户
        if current_user.is_authenticated and current_user.name == "Tom":
            return True
        return False
```

```
    # 访问页面没有权限时回调函数,可跳转登录
    def inaccessible_callback(self, name, **kwargs):
        # redirect to login page if user doesn't have access
        return redirect(url_for('myuser.login', next=request.url))
```

这样只有Tom登录之后才能看见后台管理页面:

回到顶部

五 flask-restful

快速构建restApi,还是很方便的,其实就是把一个url的增伤改查写到一个类里面:

新建api类:

```
# restful 构建api
from models import User, db

# 返回格式
resource_fields = {
    'id': fields.Integer,
    'name': fields.String,
    'age': fields.String,
    'write_time': fields.String
}

class RestUser(Resource):
    # 格式化返回结果,envelope 数据显示名字
    @marshal_with(resource_fields, envelope="user")
    def get(self, id):
        return User.query.filter_by(id=id).first(), 200
```

not running的一些坑以及解决办法(11898)

评论排行榜

1. 使用IntelliJ IDEA新建maven的javaWeb项目部署,启动访问index.jsp页面(10)

2. docker挂载war包到tomcat容器中的注意点和坑(10)

3. 个人对java中对象锁与类锁的一些理解与实例(4)

4. springcloud的Turbine配置监控多个服务的一些坑!!!!InstanceMonitor$MisconfiguredHostException,No message available","path":"/actuator/hystrix.stream,页面不显示服务或者一直loading(4)

5. Lucene7.1.0版本的索引创建与查询以及维护,包括新版本的一些新特性探索!(3)

推荐排行榜

1. vue中使用localStorage存储信息(2)

2. 个人对java中对象锁与类锁的一些理解与实例(2)

```
    def post(self, name):
        # 解析参数
        parser = reqparse.RequestParser()
        # 添加参数,校验等,help: 校验不通过时展示自定义信息, location 可指定参数解析位置
        parser.add_argument('age', type=int, required=True,
                            help='年龄必须是整数,且不为空', location='form',
                            dest='age_alis')  # dest 使用别名存储参数,原来参数名key无法获取
        args = parser.parse_args()
        # age = request.form.get('age')
        user = User(name=name, age=args['age_alis'])
        db.session.add(user)
        db.session.commit()
        # 继承父类参数
        parser_copy = parser.copy()
        # 覆盖或者删除
        parser_copy.replace_argument('age', type=int, required=True)
        parser_copy.remove_argument('age')
        return "success"
    def delete(self, id):
        user = User.query.filter_by(id=id).first()
        db.session.add(user)
        return "ok"
```

配置:

```
mail.init_app(app)
# 注册restful api
api = Api(app)
api.add_resource(RestUser, '/rest/<int:id>', '/rest/<name>')  # 可使用多个url访问接口
```

回到顶部

六 flask-admin

3. java中并发Queue种类与各自API特点以及使用场景!(2)

4. Lucene7.1.0版本的索引创建与查询以及维护,包括新版本的一些新特性探索!(1)

5. 使用IntelliJ IDEA新建maven的javaWeb项目部署,启动访问index,jsp页面(1)

Flask提供了一个扩展模块帮助我们快速搭建一个后台管理系统，这个模块就是--Flask-Admin

这个可以提供所有model数据的增删改查,而且非常灵活支持扩展,比如禁用删除,只显示某一列等等:

配置:

```
# 继承ModelView,实现一些自定义扩展
from models import User


class MyModelView(ModelView):
    # 这里可以定制权限管理
    can_delete = True
    can_create = False  # 是否能创建
    # Override displayed fields
    column_list = ('name', 'age')  # 显示的属性
    # 重写方法实现权限控制
    def is_accessible(self):
        print(current_user.name)  #  获取login的当前用户
        if current_user.is_authenticated and current_user.name == "Tom":
            return True
        return False
    # 访问页面没有权限时回调函数,可跳转登录
    def inaccessible_callback(self, name, **kwargs):
        # redirect to login page if user doesn't have access
        return redirect(url_for('myuser.login', next=request.url))
```

```
# 初始化admin后台管理
admin = Admin(app, name='MyWebSite', template_mode='bootstrap3')
# 注册模型用来管理,可自定义url,避免冲突(此处也使用蓝图,名字默认使用model的小写名字,名字不能与蓝图名字冲突)
admin.add_view(MyModelView(models.User, db.session, name=u'用户管理', url='user/manage'))
admin.add_view(UserView(name='自定义视图'))
```

定制视图(把某些页面放在后台管理上):

```
# 继承BaseView进行视图页面定制
class UserView(BaseView):
    # 使用expose进行路由,每个视图必须有一个 '/' 函数,否则报错
    @expose('/')
    def index(self):
        return self.render('boot.html')

    @expose('/user_manager')
    def user_manager(self):
        return self.render('boot.html')
```

最终效果:

« 上一篇: docker挂载war包到tomcat容器中的注意点和坑
» 下一篇: windows配置Lua开发环境