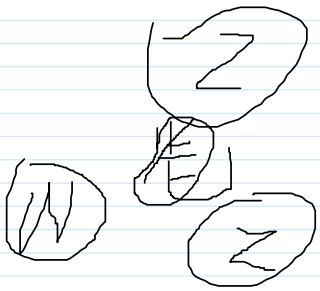
黑马比赛

2020年12月2日 21:53

DB
db admin:root/root spadmin/SPADMIN

书中自有黄金屋，大数据中自有大价值 （价值）
针对备件数据进行表述，数据量每年多少条，涉及多少个种类，价值有多少



温家宝说过： （价值）
中国有13亿人口，不管多么小的问题，只要乘以13亿，那就成为很大很大的问题

备件库存压力几个亿，不管多小的进步，只要乘以几个亿，那就成为很大很大的进步

1.增加页码
2.用深色背景
3.正文字体12号
4.逐字稿，项目背景：痛点问题，从机遇更新到痛点，解决什么问题

系统能解决多少缺点，就有多少优点

通过故事引出痛点，达成目标。

系统展示要提前

技术栈 三条 可复用 其他5个小组也可以使用我这一套架构，提高执行效率

创造价值 不是一个维度，通过快，效率高，可复用性 ，前后对比

每个页面缺少观点句

备件生命周期，通过系统截图，以及时间进度图来表示

提升效率 通过数据展示

最后加上总结页

Element-UI/VUE

2020年11月27日 12:17

1.resetFields()无效

- 表单没加ref属性
- el-form-item 没有添加prop属性
- this.nextTick(() = { this.form.name = this.name})用nextTick方法在下次DOM节点更新的时候刷新数据，
  再在关闭窗口函数里进行resetFields()

2.获取不到this指针

- 通过let that = this将指针赋值给that，在函数体内使用
- 通过() => {} 箭头函数，继承父类this指针

SQL ALCHEMY

2020年11月28日 22:03

1.in子查询

```
Object.in_(list/tuple)
```

2.having

```
group_by(year_i, year_o).\
    having(text("year_o not in ('2018','2019')")).\
    having(text('year_i < 2017')).\
```

3.SQLAlchemy实例

在__init__.py的头上 db = SQLAlchemy 实例化

在create_app()函数里 db.init_app(app)

4.model类映射表范式

```
class TmMsg(db.Model, EntityBase):
  __tablename__ = 'tm_msg'
  mid = db.Column(db.Integer, primary_key=True)
  message = db.Column(db.Text)
  uid = db.Column(db.Integer)
  create_time = db.Column(db.DateTime, default=datetime.now)
  is_read = db.Column(db.Integer, default=0)
  def __init__(self, message, uid, is_read):
    self.message = message
    self.uid = uid
    self.is_read = is_read
```

def __init__(self, message, uid, is_read):构造函数

所有数据表的实体类都需要继承SQLAlchemy的Model类

5.常用SQLAlchemy查询写法

继承Model类的数据表实体类可以通过成员函数query()进行简单查询

```
auth_user = models.AuthUser.query.filter_by(uid=au['uid']).first()
```

通过SQLAlchemy自带db.session.query查询

```
temp = db.session.query(func.year(tsp.o_warehouse_date), tsp.sno).\
        filter(func.year(tsp.o_warehouse_date) == start_year).\
            group_by(func.year(tsp.o_warehouse_date), tsp.sno).\
              having(func.count(distinct(tsp.asset_no)) == 10).\
                  order_by(desc(func.sum(tsp.amount))).limit(5).all()
```

返回的数据为TmSparePart数据表类的数组

6.常用SQLAlchemy的Config

```
app.config['SQLALCHEMY_DATABASE_URI'] = "mysql+pymysql://<用户名>:<密码>@<hostname:port>/dbschema"
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
app.config['SCHEDULER_API_ENABLED'] = True
app.config['SCHEDULER_TIMEZONE'] = 'Asia/Shanghai'
app.config['SCHEDULER_JOBSTORES'] = {'default': SQLAlchemyJobStore(url=app.config['SQLALCHEMY_DATABASE_URI'])}
```

Python

2020年12月3日 17:43

1.获取文件所在地址

```
os.path.dirname(__file__)
```

2.获取当前APP实例以及获取当前app上下文

```
db_config = current_app.config.get('SQLALCHEMY_DATABASE_URI')
print(db_config)
with current_app.app_context():
    app = current_app
    db_engine = db.get_engine(app=app)
    print(db_engine)
```

Flask

2020年12月14日 14:18

1.搭建Flask应用建议使用工厂设计模式

```
def create_app(test_config=None):
    app = Flask(__name__, instance_relative_config=True)
        app.jinja_env.variable_start_string = '[['
    app.jinja_env.variable_end_string = ']]'
    if test_config is None:
        app.config.from_pyfile('config.py',silent=True)
    else:
        app.config.from_mapping(test_config)
    try:
        os.mkdir(app.instance_path)
    except OSError:
        pass

        # app.add_url_rule('/',endpoint='index')
        # 这里可以注册blueprint
        return app
```

2.Flask启动命令

```
windows cmd:
set flask_app=<flask所在目录>
set flask_env = dvelopment
flask run
```

3.获取FORM post的数据

```
from flask import request
parameter = request.form['parameter']
```

Pandas

2020年12月3日 16:54

1.删除列

```
df.drop(['day','month','year'],axis=1,inplace=True)
```

2.打印显示所有列

```
pd.set_option('display.max_columns',None)
```

3.df.groupby完整示例

```
dfn = df.groupby([df['o_warehouse_date'].apply(lambda x:x.year),'sno'],as_index=False).ag
g({'asset_no':pd.Series.nunique,'amount':np.sum}).sort_values('sno',ascending=False)
```

nunique:唯一

4.更改列属性

```
df['column'].astype(str)
df['column'].map(str)
```

5.column值匹配list

```
temp = ['PFA1','PFA2','PFA3','PFE','PFN','PFY','PFS','PFH','PFC','PFW']
df['asset_no'].isin(temp)
```

6.dataframe转换dict

```
df.to_dict(orient = 'list')
```

- orient = ‘dict’ ，是函数默认的，转化后的字典形式：{column(列名) : {index(行名) : value(值))}};
- orient = ‘list’ ，转化后的字典形式：{column(列名) :{[values](值)}};
- orient = ‘series’ ，转化后的字典形式：{column(列名) : Series (values) (值)};
- orient = ‘split’ ，转化后的字典形式：{ ‘index’ : [index], ‘columns’ :[columns], ’ data ‘ : [values]};
- orient = ‘records’ ，转化后是 list形式：[{column(列名) : value(值)} …{column:value}];
- orient = ‘index’ ，转化后的字典形式：{index(值) : {column(列名) : value(值)}};

7.合并两个dataframe

```
df_n = pd.concat([a_df, b_df], ignore_index=True)
```

传递df数组，通过ignore_index参数来控制

8.重命名列名

```
df. rename (columns={’ ori_name’ :’ tar_name’ })
```

设计模式

2020年12月3日 17:43

抽象类不能实例化，只能被继承。
抽象类为代码服用而生，子类可以通过继承抽象类中的属性和方法避免子类种重复编写相同的代码。