

DAG的可视化

为了给用户友好的界面显示效果，在网页往往需要显示出DAG的图形。
使用echarts可以很好的完成这个功能。

<http://echarts.baidu.com/examples/editor.html?c=graph-simple>

在线修改例子如下

```
option = {
  title: {
    text: 'DAG 简单示例 Echarts'
  },
  tooltip: {},
  animationDurationUpdate: 1500,
  animationEasingUpdate: 'quinticInOut',
  series : [
    {
      type: 'graph',
      layout: 'none',
      symbolSize: 50,
      roam: true,
      label: {
        normal: {
          show: true
        }
      },
      edgeSymbol: ['circle', 'arrow'],
      edgeSymbolSize: [4, 10],
      edgeLabel: {
        normal: {
          textStyle: {
            fontSize: 20
          }
        }
      },
      data: [{
        name: 'A',
        x: 300,
        y: 300
      }, {
        name: 'B',
        x: 400,
        y: 300
      }, {
        name: 'C',
        x: 330,
        y: 360
      }, {
        name: 'D',
        x: 380,
```

```

        y: 330
      }],
      // links: [],
      links: [{
        source: 0,
        target: 1
      }, {
        source: 0,
        target: 2
      }, {
        source: 2,
        target: 1
      }, {
        source: 1,
        target: 3
      }],
      lineStyle: {
        normal: {
          opacity: 0.9,
          width: 2,
          curveness: 0
        }
      }
    }
  ]
};

```

title 标题，对象

legend 图例

xAxis X轴，对象，data属性就是x轴数据

yAxis Y轴，对象，type设定数据类型，'value'是值类型

series 数据序列，数组，每一个数组元素是一个对象。

获得效果如下

DAG 简单示例 Echarts



Flask框架代码实现

- 使用Flask微框架
- 使用Jinja2模板技术
- 使用jQuery发起AJAX异步调用
- 使用ECharts图表组件
- 使用uWSGI部署

在pycharm项目新建项目，准备编写Flask项目。

Flask安装

```
$ pip install flask
```

编程快速入门 <http://docs.jinkan.org/docs/flask/quickstart.html#quickstart>

在项目根目录下构建3个目录

web目录，存放代码

templates目录，存放模板文件

static目录存放js、css等静态文件。其下建立js目录，放入jquery、echarts的js文件

模板定义

准备4个模板文件，如下

index.html 首页

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">

<html>
<head>
    <title>马哥教育</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
</head>
<body>
<h2>马哥教育流程系统-Flask-JQuery-Ajax-ECharts测试</h2>
<ur>
    <li><a href="1">图表1</a></li>
    <li><a href="2">图表2</a></li>
    <li><a href="3">图表3</a></li>
</ur>
</body>
</html>
```

chart1.html 简单图表

注意：模板中调用url_for生成路径，这点很重要，否则访问静态路径会出错（404）

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>ECharts</title>
    <!-- 引入 jquery -->
    <script src={{url_for('static', filename="js/jquery-2.1.1.min.js")}}></script>
    <!-- 引入 echarts.js -->
    <script src={{url_for('static', filename="js/echarts.min.js")}}></script>
</head>
<body>
    <!-- 为ECharts准备一个具备大小（宽高）的Dom -->
    <div id="main" style="width: 600px; height: 400px;"></div>
    <script type="text/javascript">
        // 基于准备好的dom，初始化echarts实例
        var myChart = echarts.init(document.getElementById('main'));
        // JQuery Ajax调用
        $.get('/dag/1', function(data){
            console.log(data);
            // 指定图表的配置项和数据
            var option = {
                title: {
                    text: 'ECharts 入门示例'
                },
                tooltip: {},
                legend: { // 图例
                    data: ['销量', '产量']
                },
                xAxis: { // x轴
```

```

        data: data.xs
    },
    yAxis: {type: 'value'}, // Y轴
    series: [{ // 数据数据
        name: '产量',
        type: 'bar',
        data: data.data
    },
    {
        name: '销量',
        type: 'bar',
        data: data.data.map(x=>x + parseInt(Math.random()*10 - 5))
    }
    ]
};

// 使用刚指定的配置项和数据显示图表
myChart.setOption(option);
})
</script>
</body>
</html>

```

chart2.html DAG图表 (JQuery AJAX)

```

<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>ECharts</title>
    <!-- 引入 jquery -->
    <script src={{url_for('static', filename="js/jquery-2.1.1.min.js")}}></script>
    <!-- 引入 echarts.js -->
    <script src={{url_for('static', filename="js/echarts.min.js")}}></script>
</head>
<body>
    <!-- 为ECharts准备一个具备大小 ( 宽高 ) 的Dom -->
    <div id="main" style="width: 600px;height:400px;"></div>
    <script type="text/javascript">
        // 基于准备好的dom, 初始化echarts实例
        var myChart = echarts.init(document.getElementById('main'));

        $.get('/dag/2', function(data){
            console.log(data);
            // 指定图表的配置项和数据
            option = {
                title: {
                    text: 'DAG 简单示例 Echarts'
                },
                tooltip: {},
                animationDurationUpdate: 1500,
                animationEasingUpdate: 'quinticInOut',
                series : [

```

```

    {
      type: 'graph',
      layout: 'none',
      symbolSize: 50,
      roam: true,
      label: {
        normal: {
          show: true
        }
      },
      edgeSymbol: ['circle', 'arrow'],
      edgeSymbolSize: [4, 10],
      edgeLabel: {
        normal: {
          textStyle: {
            fontSize: 20
          }
        }
      },
      data: data.data,
      // links: [],
      links: data.links,
      lineStyle: {
        normal: {
          opacity: 0.9,
          width: 2,
          curveness: 0
        }
      },
      tooltip: { // 提示框，鼠标放在节点或边上试一试
        formatter: "{b}<br />{c}", // {b}表示类目，{c}表示数值
        backgroundColor: "#000000" //背景色
      }
    }
  ]
};

// 使用刚指定的配置项和数据显示图表
myChart.setOption(option);
})
</script>
</body>
</html>

```

chart3.html DAG图表 (JQuery AJAX)

```

<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>ECharts</title>
  <!-- 引入 jquery -->

```

```

<script src={{url_for('static', filename="js/jquery-2.1.1.min.js")}}></script>
<!-- 引入 echarts.js -->
<script src={{url_for('static', filename="js/echarts.min.js")}}></script>
</head>
<body>
    <!-- 为ECharts准备一个具备大小 ( 宽高 ) 的Dom -->
    <div id="main" style="width: 600px;height:400px;"></div>
    <script type="text/javascript">
        // 基于准备好的dom, 初始化echarts实例
        var myChart = echarts.init(document.getElementById('main'));

        $.get('/dag/3', function(data){
            console.log(data);
            myChart.hideLoading();
            // 指定图表的配置项和数据
            option = {
                title: {
                    text: data.title
                },
                tooltip: { trigger: 'item' },
                animationDurationUpdate: 1500,
                animationEasingUpdate: 'quinticInOut',
                series : [
                    {
                        type: 'graph',
                        layout: 'none',
                        symbolSize: 50,
                        roam: true,
                        label: {
                            normal: {
                                show: true
                            }
                        },
                        edgeSymbol: ['circle', 'arrow'],
                        edgeSymbolSize: [4, 10],
                        edgeLabel: {
                            normal: {
                                textStyle: {
                                    fontSize: 20
                                }
                            }
                        },
                        data: data.data,
                        // links: [],
                        links: data.links,
                        lineStyle: {
                            show: false,
                            normal: {
                                opacity: 0.9,
                                width: 2,
                                curveness: 0
                            }
                        },
                    },
                ],
            },
        });
    </script>

```

```

        tooltip: { // 提示框，鼠标放在节点或边上试一试
            // 使用函数重新定义显示文字的格式，回调送入3个参数
            formatter: function (params, ticket, callback) {
                if (params.dataType === 'edge') // 连线没有值返回空串
                    return '';
                if (params.value)
                    return params.name + '<br />' + params.value
                return params.name
            }
            //, backgroundColor: "#000000"
        }
    }
}

};

// 使用刚指定的配置项和数据显示图表
myChart.setOption(option);

// 遍历数据
echarts.util.map(data.data, function(item, dataIndex){
    console.log(item);
    console.log(dataIndex);
});

// 鼠标事件，click点击
myChart.on('mouseover', function (item) {
    console.log(item);
    if (item.value) {
        console.log(item.value)
    }
});

});
</script>
</body>
</html>

```

服务端代码实现

创建应用app


```

# web/__init__.py中
from flask import Flask, make_response, render_template, jsonify

app = Flask('pipeline_web')

@app.route('/', methods=['GET']) # 路由, 可以指定方法列表, 缺省GET
def index(): # 视图函数
    return render_template('index.html')

@app.route('/<int:graphid>') # index.html中访问不同的模板页
def showdag(graphid):
    return render_template('chart{}.html'.format(graphid))

```

在项目根目录创建测试文件

```

# 启动测试应用
if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)

```

模板一旦被调用, 返回的HTML页面会立即发起AJAX调用, 请求DAG数据, 视图函数会向Service层请求数据。

从pipeline中复制config.py、model.py到web目录下

在web包下创建service.py文件

```

from .model import getdb, Pipeline, Track, Vertex, Edge
import random

db = getdb() # WEB项目和Pipeline项目最好分开使用连接和session

def randomxy():
    return random.randint(300, 500) # 随机模仿 x y坐标

def getdag(p_id): # 根据pipeline的id返回流程数据, 让前端页面绘制DAG图
    ps = db.session.query(
        Pipeline.id, Pipeline.name, Pipeline.state,
        Vertex.id, Vertex.name, Vertex.script, Track.script
    ).join(Track, (Pipeline.id == Track.p_id) & (Pipeline.id == 1))\
    .join(Vertex, Vertex.id == Track.v_id)

    edges = db.session.query(Edge.tail, Edge.head).\
    join(Pipeline, (Pipeline.g_id == Edge.g_id) & (Pipeline.id == 1))

    data = [] # 顶点数据
    vertexes = {} # 让edges查询少join
    title = ''
    for p_id, p_name, p_state, v_id, v_name, v_script, t_script in ps:
        if not title:
            title = p_name

```

```

        data.append({
            'name': v_name,
            'x': randomxy(),
            'y': randomxy(),
            'value': t_script if t_script else v_script
        })
        vertexes[v_id] = v_name
    print(data)

    links = [] # 边
    for tail, head in edges:
        links.append({
            'source' : vertexes[tail],
            'target' : vertexes[head]
        })

    return {'title':title, 'data':data, 'links':links}

```

完成所有剩余代码

```

# web/__init__.py中
from flask import Flask, make_response, render_template, jsonify
from .service import getdag

app = Flask('pipeline_web')

@app.route('/', methods=['GET']) # 路由, 可以指定方法列表, 缺省GET
def index(): # 视图函数
    return render_template('index.html')

@app.route('/<int:graphid>') # index.html中访问不同的模板页
def showdag(graphid):
    return render_template('chart{}.html'.format(graphid))

@app.route('/dag/<int:graphid>') # (rule, **options):
def showjaxdag(graphid):
    if graphid == 1:
        return simplegraph()
    elif graphid == 2:
        return jsonify(getdag(1))
    elif graphid == 3:
        return jsonify(getdag(1))

def simplegraph():
    xs = ["衬衫", "羊毛衫", "雪纺衫", "裤子", "高跟鞋", "袜子"]
    data = [5, 20, 36, 10, 10, 20]
    return jsonify({'xs': xs, 'data': data})

```

启动服务，看看效果。

uWSGI + Flask部署

uwsgi安装在Linux服务器上

```
# yum install python-devel  
  
$ pip install uwsgi flask  
$ pip install pymysql sqlalchemy
```

在服务器构建Python运行虚拟环境，建立目录，将templates、static、web三个目录及文件复制到服务器上该目录下。

在Flask项目根目录下运行 `$ uwsgi --http :8000 -w web:app`

uwsgi的配置文件

```
[uwsgi]  
http = 0.0.0.0:5000  
module = web:app
```

```
$ uwsgi flask.ini
```

