

web编程期末项目

web编程期末项目

Demo

代码结构

功能实现

1. 用户可注册登录网站，非注册用户不可登录查看数据及停用启用用户实现

前端实现

实现登陆操作（停用判断）

angular代码

路由代码

1) 基于dao实现

2) 直接与mysql交互（async和await处理）

实现注册操作

angular代码

路由代码

1) 基于DAO实现

2) 直接与mysql交互

非注册用户不能登陆实现

停用启用用户前端实现

启用用户实现

路由代码

停用用户实现

路由代码

DAO层实现

2. 用户注册、登录、查询等操作记入数据库中的日志

借助 Express 框架记录日志的中间件 morgan

3. 爬虫数据查询结果列表支持分页和排序

主页面前端代码

angularJS显示与隐藏实现细节

实现布尔表达式

1) 当点击“检索”按钮—>调用 show_search() 函数

2) 显示输入框

3) 调用 search() 函数

4) 传给路由

4. 用Echarts或者D3实现3个以上的数据分析图表展示在网站中

以饼图为例

词云图

柱状图

折线图

5. 实现一个管理端界面，可以查看（查看用户的操作记录）和管理（停用启用）注册用户。

资质审核

前端点按时调用check函数

调用 check() 函数

路由代码

查看操作日志

前端
调用searchlog函数
路由代码
logDAO.js
停用启用
6.实现对爬虫数据中文分词的查询
 分词处理
 分词查询
7.实现查询结果按照主题词打分的排序
 打分机制（TF-IDF）
 后端查询语句
相关错误和问题处理
1.js报404
2.Unknown column 'xx' in 'where clause'
3.cannot read property ** of undefined
4.Truncated incorrect DOUBLE value
5.Cannot set headers after they are sent to the client
6.出现mysql查询正常但是无法写入的情况
7.分词效果不理想
总结

Demo

<https://www.bilibili.com/video/BV1f44y1i7tM/>

代码结构

final_project 2

```
├── .DS_Store
├── .idea
|   ├── final_project.iml
|   ├── jsLibraryMappings.xml
|   ├── misc.xml
|   ├── modules.xml
|   └── runConfigurations
    |       └── Debug.xml
    └── workspace.xml
└── app.js
└── bin
    └── www
```

```
|── conf
|   └── mysqlConf.js
├── createtables.sql
├── dao
|   ├── logDAO.js
|   ├── newsDAO.js
|   ├── scu_stopwords.txt
|   ├── userDAO.js
|   └── userSqlMap.js
├── package-lock.json
├── package.json
└── public
    ├── .DS_Store
    ├── admin.html
    ├── images
    |   ├── .DS_Store
    |   ├── background.jpg
    |   ├── logo.png
    |   └── logo1.png
    ├── index.html
    ├── index1.html
    ├── javascripts
    |   ├── .DS_Store
    |   ├── dist
    |   |   ├── echarts-wordcloud.js
    |   |   └── echarts-wordcloud.min.js
    |   ├── index.js
    |   ├── mysql.js
    |   ├── news.js
    |   ├── rank.js
    |   └── scu_stopwords.txt
```

```
|  |    └── stopword.js  
|  └── news.html  
|  └── search.html  
|  └── searchlog.html  
|  └── stylesheets  
|    └── index.css  
└── routes  
  |  └── .DS_Store  
  |  └── freqchange.js  
  |  └── news.js  
  |  └── scu_stopwords.txt  
  |  └── users.js  
  |    └── wordcut.js  
└── views  
  ├── error.pug  
  ├── index.pug  
  └── layout.pug
```

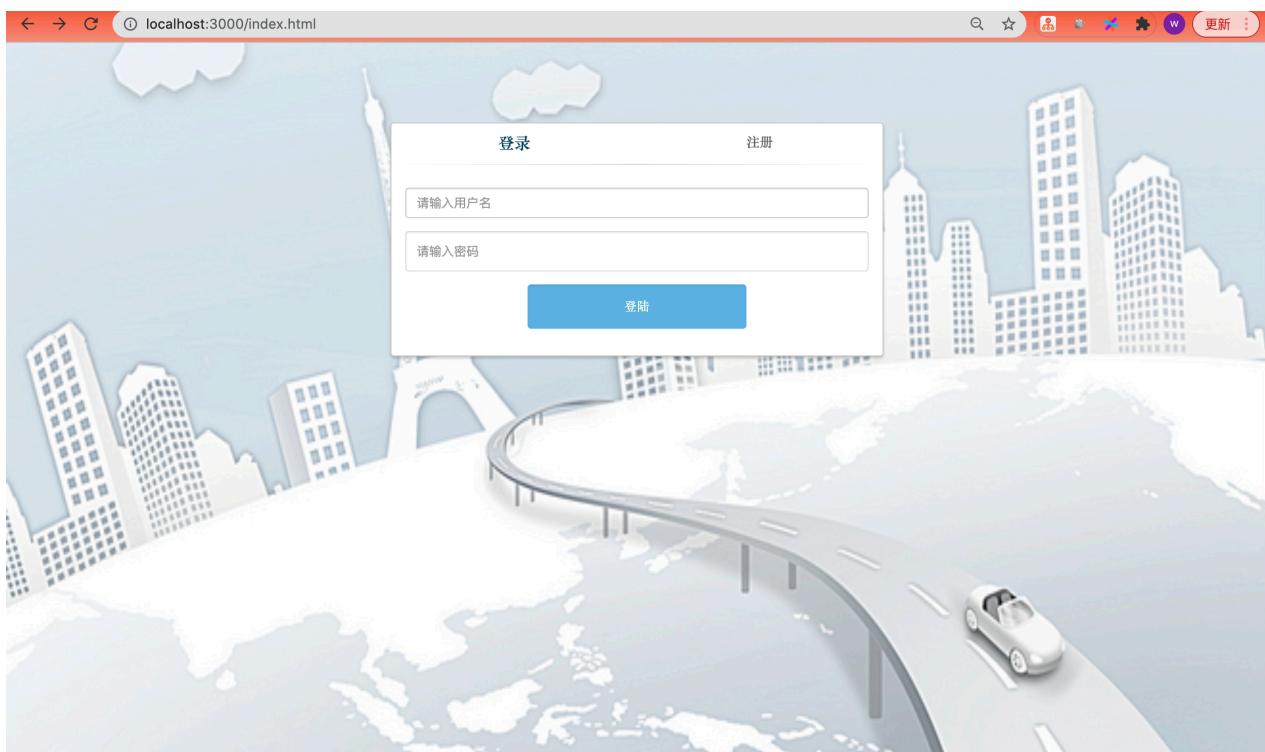
...

功能实现

1. 用户可注册登录网站，非注册用户不可登录查看数据及停用启用用户实现

基于dao和直接的mysql数据库交互实现了两种方式的登陆注册

界面效果如下



前端实现

index.html

登陆界面

首先在根元素上定义 AngularJS 应用“login”，指定AngularJS应用程序管理的边界，使得在ng-app内部的指令起作用,然后设置相应的登陆窗口。通过 ng-model 指令把输入域的用户名和密码以 “username” 和 “password” 绑定到当前作用域中，以便 \$scope 对象在控制器和视图之间传递数据；当按钮事件被触发时，调用 check_pwd() 函数检查密码是否正确。

```
<div class="container" ng-controller="loginCtrl">
  <div class="row">
    <div class="col-md-6 col-md-offset-3">
      <div class="panel panel-login">
        <div class="panel-heading">
          <div class="row">
            <div class="col-xs-6">
              <a href="#" class="active" id="login-form-link">登
              录</a>
            </div>
            <div class="col-xs-6">
              <a href="#" id="register-form-link">注册</a>
            </div>
          </div>
          <hr>
        </div>
        <div class="panel-body">
          <div class="row">
            <div class="col-lg-12">
```

```
<form id="login-form" method="post" role="form"
style="display: block;">
<!-- 登陆部分-->
<div class="form-group">
    <input ng-model="username" tabindex="1"
class="form-control" placeholder="请输入用户名" value="" />
</div>
<div class="form-group">
    <input type="password" ng-model="password"
tabindex="2" class="form-control" placeholder="请输入密码">
</div>
<!-->
<!-->
<div class="form-group text-center">-->
<input type="checkbox" tabindex="3"
class="" name="remember" id="remember">-->
<!-->
<label for="remember"> Remember
Me</label>-->
<!-->

<div class="form-group">
    <div class="row">
        <div class="col-sm-6 col-sm-offset-3">
            <button id="login-submit"
tabindex="4" class="form-control btn btn-login" ng-click="check_pwd()">登
陆</button>
        </div>
    </div>
</div>
</form>
<form id="register-form" method="post" role="form"
style="display: none;">
    <div class="form-group">
        <input ng-model="add_username" tabindex="1"
class="form-control" placeholder="请输入用户名" value="" />
    </div>

    <div class="form-group">
        <input type="password" ng-
model="add_password" tabindex="2" class="form-control" placeholder="请输入密码">
    </div>

    <div class="form-group">
        <input type="password" ng-
model="confirm_password" tabindex="2" class="form-control" placeholder="请确认您
刚输入的密码">
    </div>
    <div class="form-group">
        <div class="row">
            <div class="col-sm-6 col-sm-offset-3">
```

```

        <button tabindex="4" class="form-control btn btn-register" ng-click="doAdd()">注册</button>
    </div>
    </div>
    </div>

    </form>

```

实现登陆操作（停用判断）

功能实现——对用户有相应的提示：

- 提示用户名或密码输错
- 用户不存在
- 用户被停用
- 登陆成功

angular代码

先定义控制器“loginCtrl”：初始化视图中输入的数据，作为存储数据的容器；同时通过 \$scope 对象把函数数行为暴露给视图；监视模型的变化，做出相应的逻辑处理。

```

var app = angular.module('login', []);
app.controller('loginCtrl', function ($scope, $http, $timeout) {

    // 登录时，检查用户输入的账户密码是否与数据库中的一致
    $scope.check_pwd = function () {
        var data = JSON.stringify({
            username: $scope.username,
            password: $scope.password
        });
        $http.post("/users/login", data)
            .then(
                function (res) {
                    if(res.data.msg=='ok') {
                        window.alert("登陆成功，欢迎～")
                        window.location.href='/news.html';
                    }else{
                        $scope.msg=res.data.msg;
                        window.alert($scope.msg);
                    }
                },
                function (err) {
                    $scope.msg = err.data;
                    window.alert($scope.msg);
                }
            );
    };
});

```

```
};
```

通过window.alert进行弹窗提醒，将message传过来的信息进行相应的显示。

如果登陆成功，跳转到网站查询首页

路由代码

Users.js

将用户名传入数据库获取对应的密码

如果返回的用户名长度为零，说明用户不存在；

如果用户的stopflag为1表明已被停用则不能注册。如果获取到密码，检查和用户输入的密码是否匹配：若匹配则发送 json 响应，msg 赋值为“ok”，并且把用户名保存到服务器内存中，以便记录用户登录操作日志和检查用户对页面进行操作时的登录状态；否则 msg 为错误提示‘用户名或密码错误！请检查输入是否有误’。

1) 基于dao实现

```
router.post('/login', function(req, res) {
  var username = req.body.username;
  var password = req.body.password;
  // var sess = req.session;

  userDAO.getByUsername(username, function (user) {
    if(user.length==0){
      res.json({msg: '用户不存在！请先注册'});
    }else {
      console.log("stop",user[0].stopflag)
      if(user[0].stopflag==1){
        res.json({msg: '对不起，用户已被停用'});
      }
      if(password==user[0].password){
        req.session['username'] = username;
        res.cookie('username', username);
        res.json({msg: 'ok'});
        // res.json({msg:'ok'});
      }else{
        res.json({msg: '用户名或密码错误！请检查输入是否有误'});
      }
    }
  });
});
```

2) 直接与mysql交互（async和await处理）

功能实现思路类似第一种，不过这里进行了async 和await。

async是异步的意思，看上去是一个异步标识，表示这个函数中有异步执行的代码。

await就是等待 **async**执行完，才会执行后面的东西，等待的东西是异步的，它就会阻塞当前代码。它只能在**async**函数里使用，虽然阻塞，但是是异步的。

```
router.post('/login', async function (req, res) {
  username = req.body.name;
  password = req.body.passwd;
  console.log(username);
  var response = {};
  var fetch_name_Sql = "select passwd from UserInfo where name = '" +username+
"';";
  console.log(fetch_name_Sql);
  mysql.query(fetch_name_Sql, async function(err, result) {
    // console.log(result);
    if (err) {
      console.log(err)
    } else {

      if(username != '' && password != ''){
        console.log(username)
        console.log(password)
        var select_Sql = "select passwd from UserInfo where name ='" +
username + "';";
        let value = await mysql.promise_query(select_Sql, function () {});

        var select_Sql1 = "select * from UserInfo where name ='" + username +
"';";
        let value1 = await mysql.promise_query(select_Sql1, function () {});
        console.log("value1",value1)
        if(value1.length==0){
          console.log("用户未注册");
          response.status = '用户未注册';
          console.log('not regist');
          console.log(response);
          res.send(JSON.stringify(response));
          return;
        }else{
          if(value[0].passwd === password){
            response.status = '登陆成功~';
            var insert_sql = "insert into Log(username,operation) values(\\""+
username + "\','" + "'login')");
            let value = await mysql.promise_query(insert_sql, function () {
});

            res.cookie("username", username);
            console.log(response);
            res.send(JSON.stringify(response));
          }
        }
      }
    }
  })
})
```

```

    }else{
        response.status = '密码错误, 请重试';
        console.log('passwd error');
        console.log(response);
        res.send(JSON.stringify(response));
        return ;
    }

}

}else{
    response.status = '请输入用户名和密码~';

    console.log(response);
    res.send(JSON.stringify(response));
    // rets.send('500')
}
}
)
}
)
}

```

实现注册操作

angular代码

先检验两次输入密码是否一致，如果不一致，提示'两次密码不一致!'，注册成功则进行相应跳转，否则返回错误信息

```

$scope.doAdd = function () {
    // 检查用户注册时，输入的两次密码是否一致
    if($scope.add_password!==$scope.confirm_password){
        // $timeout(function () {
        //     $scope.msg = '两次密码不一致!';
        // },100);
        $scope.msg = '两次密码不一致!';
        window.alert('两次密码不一致!')
    }
    else {
        var data = JSON.stringify({
            username: $scope.add_username,
            password: $scope.add_password
        });
        $http.post("/users/register", data)
        .then(function (res) {
            if(res.data.msg=='成功注册! 请登录') {
                window.alert('注册成功, 请登录')
                $scope.msg=res.data.msg;
                $timeout(function () {
                    window.location.href='index.html';
                },2000);
            }
        })
    }
}

```

```

        } else {

            $scope.msg = res.data.msg;
            window.alert($scope.msg)
        }
    }, function (err) {
        $scope.msg = err.data;
    });
}
);
}
);

```

路由代码

先检查用户是否已存在，如果已存在，则无需再添加用户信息

1) 基于DAO实现

```

router.post('/register', function (req, res) {
    var add_user = req.body;
    // 先检查用户是否存在
    userDAO.getByUsername(add_user.username, function (user) {
        if (user.length != 0) {
            // res.render('index', {msg:'用户不存在!'});
            res.json({msg: '用户已存在!'});

        }else {
            userDAO.add(add_user, function (success) {
                res.json({msg: '成功注册! 请登录'});
            })
        }
    });
});

```

2) 直接与mysql交互

```

router.post('/regist', async function (req, res) {
    response = {};
    username = req.body.name;
    password = req.body.passwd;
    console.log(username);

    var fetch_name_Sql = "select name from UserInfo where name ='" +username +
    "';";
    mysql.query(fetch_name_Sql, async function(err, result) {
        if (err) {
            console.log(err)
        } else {

```

```

let value = await mysql.promise_query(fetch_name_Sql, function () {});
console.log(value)
if(value.length!=0) {
    console.log("用户已注册")

    response.status = "用户已注册";
    console.log(response);
    res.end(JSON.stringify(response));
    return ;
} else {
    var fetchAddSql = "insert into UserInfo (name, passwd) values('" +
username + "','" + password+"')";

    mysql.query(fetchAddSql, function(err, result) {
        if(err) {
            console.log(err);
            oError.innerHTML = "用户注册失败";
            response.status = "用户注册失败";
            console.log(response);
            res.end(JSON.stringify(response));
            return ;
        } else {
            var fetchInsertSql="insert into Log(username,operation)
values('" + username + "','" + 'register')"

            mysql.query(fetchInsertSql, function(err,result) {
                if(err) {
                    console.log(err);
                }
            })
        }
    });
    response.status = true;
    console.log(response);
    res.end(JSON.stringify(response));
}
}
);
}
}

```

非注册用户不能登陆实现

在每个主页显示内容时进行判断

路由代码(以news.js为例)

判断用户名是否为undefined, 跳转到注册界面

```

if (request.session['username'] === undefined) {
    // response.redirect('/index.html')
    response.json({message: 'url', result: '/index.html'});
}

```

停用启用用户前端实现

对应调用/users/start以及/users/stop进行相应操作

```

<!DOCTYPE html>
<html ng-app="login">
<head>
    <meta charset="utf-8" />
    <title>用户管理</title>
    <link rel="stylesheet"
    href="http://cdn.bootcss.com/bootstrap/3.3.0/css/bootstrap.min.css">
        <script src="https://cdn.staticfile.org/jquery/3.2.1/jquery.min.js">
    </script>
        <script
src="https://cdn.staticfile.org/popper.js/1.12.5/umd/popper.min.js"></script>
        <script src="https://cdn.staticfile.org/twitter-
bootstrap/4.1.0/js/bootstrap.min.js"></script>
<!--    <script src="../node_modules/angular/angular.min.js"></script>-->
        <script src="/angular/angular.min.js"></script>

<!--    引入自己的样式与js-->
    <link rel="stylesheet" type="text/css" href="stylesheets/index.css">
    <script type="text/javascript" src="javascripts/index.js"></script>
    <script>
        var app = angular.module('login', []);
        app.controller('loginCtrl', function ($scope, $http, $timeout) {

            // 登录时，检查用户输入的账户密码是否与数据库中的一致
            $scope.start = function () {
                var data = JSON.stringify({
                    username: $scope.add_username
                });
                $http.post("/users/start", data)
                    .then(
                        function (res) {
                            if(res.data.msg=='启用用户成功!') {
                                window.alert("启用用户成功!")
                                window.location.href='/news.html';
                            }else{
                                $scope.msg=res.data.msg;
                                window.alert($scope.msg);
                            }
                        }
                    )
            }
        })
    </script>

```

```

        },
        function (err) {
            $scope.msg = err.data;
            window.alert($scope.msg);
        });
    };

//增加注册用户
$scope.stop = function () {
    var data = JSON.stringify({
        username: $scope.username
    });
    $http.post("/users/stop", data)
        .then(
            function (res) {
                if(res.data.msg=='停用用户成功! ') {
                    window.alert("停用用户成功!")
                    window.location.href='/news.html';
                }else{
                    $scope.msg=res.data.msg;
                    window.alert($scope.msg);
                }
            },
            function (err) {
                $scope.msg = err.data;
                window.alert($scope.msg);
            });
};

});

```

</script>

</head>

<body>

<div class="container" ng-controller="loginCtrl">

<div class="row">

<div class="col-md-6 col-md-offset-3">

<div class="panel panel-login">

<div class="panel-heading">

<div class="row">

<div class="col-xs-6">

停用

</div>

<div class="col-xs-6">

启用

</div>

<hr>

</div>

<div class="panel-body">

```

<div class="row">
    <div class="col-lg-12">
        <form id="login-form" method="post" role="form"
style="display: block;">
    <!-- 登陆部分-->
        <div class="form-group">
            <input ng-model="username" tabindex="1"
class="form-control" placeholder="请输入用户名" value="" />
        </div>
    <!-- -->
    <!-- -->
        <input type="checkbox" tabindex="3"
class="" name="remember" id="remember">-->
    <!-- -->
        <label for="remember"> Remember
Me</label>-->
    <!-- -->

        <div class="form-group">
            <div class="row">
                <div class="col-sm-6 col-sm-offset-3">
                    <button id="login-submit"
tabindex="4" class="form-control btn btn-login" ng-click="stop()">停用</button>
                </div>
            </div>
        </div>
    </form>
    <form id="register-form" method="post" role="form"
style="display: none;">
        <div class="form-group">
            <input ng-model="add_username" tabindex="1"
class="form-control" placeholder="请输入用户名" value="" />
        </div>
        <div class="form-group">
            <div class="row">
                <div class="col-sm-6 col-sm-offset-3">
                    <button tabindex="4" class="form-
control btn btn-register" ng-click="start()">启用</button>
                </div>
            </div>
        </div>
    </form>
</div>
</div>
</div>

```

启用用户实现

路由代码

先检查用户是否存在

如果用户stopflag已经为0表明已经启用，无需重复启用同一用户

否则则返回msg'启用用户成功！'

```
router.post('/start', function (req, res) {
  var start_user = req.body;
  // 先检查用户是否存在
  console.log(start_user);
  userDAO.getByUsername(start_user.username, function (user) {
    console.log("user", user[0]);
    if (user[0].stopflag == 0) {
      // res.render('index', {msg: '用户不存在！'});
      res.json({msg: '用户已为启用状态，无需重复启用！'});
    } else {
      userDAO.start(start_user.username, function (success) {
        res.json({msg: '启用用户成功！'})
      })
    }
  });
});
```

停用用户实现

路由代码

先检查用户是否存在

如果用户stopflag已经为1表明已经停用，无需重复停用同一用户

否则则返回msg'停用用户成功！'

```
router.post('/stop', function (req, res) {
  var stop_user = req.body;
  // 先检查用户是否存在
  userDAO.getByUsername(stop_user.username, function (user) {
    console.log("user", user[0])
    console.log(user[0].stopflag)
    if (user[0].stopflag == 1) {
      // res.render('index', {msg: '用户不存在！'});
      res.json({msg: '用户已停用，无需重复停用！'});
    } else {
      userDAO.stop(stop_user.username, function (success) {
        res.json({msg: '停用用户成功！'})
      })
    }
});
```

```
});
```

```
});
```

DAO层实现

使用了连接池，重复使用数据库连接，而不必每执行一次CRUD操作就获取、释放一次数据库连接，从而提高了对数据库操作的性能。

add:负责将用户名和密码添加到用户的表中

getByUsername:通过用户名获取密码/用户停用标志

stop:将用户停用， stopflag设置为1

start:将用户启用， stopflag设置为0

```
module.exports = {
    add: function (user, callback) {
        pool.query(userSqlMap.add, [user.username, user.password], function (error, result) {
            if (error) throw error;
            callback(result.affectedRows > 0);
        });
    },
    getUsername: function (username, callback) {
        pool.query(userSqlMap.getByUsername, [username], function (error, result) {
            if (error) throw error;
            callback(result);
        });
    },
    stop:function (username, callback) {
        pool.query(userSqlMap.stop, [username], function (error, result) {
            console.log(userSqlMap.stop)
            console.log(username)
            if (error) throw error;
            callback(result);
        });
    },
    start:function (username, callback) {
        pool.query(userSqlMap.start, [username], function (error, result) {
            console.log(userSqlMap.start)
            console.log(username)
            if (error) throw error;
            callback(result);
        });
    }
};
```

定义userSqlMap写明对应的mysql语句

```
var userSqlMap = {
  add: 'insert into user(username, password) values(?, ?)', //注册时用
  getByUsername: 'select username, password,stopflag from user where username = ? ', //登陆时用
  stop: 'update user set stopflag = 1 where username = ? ', //停用用户
  start: 'update user set stopflag = 0 where username = ? ' //启用用户
};
```

2. 用户注册、登录、查询等操作记入数据库中的日志

借助 Express 框架记录日志的中间件 morgan

在 app.js 文件中引入了该中间件 `var logger = require('morgan');`

设置 session 的有效时间

打印对应日志信息，username为已登陆用户名和notlogin

```
//设置session
app.use(session({
  secret: 'sessiontest', //与cookieParser中的一致
  resave: true,
  saveUninitialized: false, // 是否保存未初始化的会话
  cookie : {
    maxAge : 1000 * 60 * 60, // 设置 session 的有效时间, 单位毫秒
  },
}));

let method = '';
app.use(logger(function (tokens, req, res) {
  console.log('打印的日志信息: ');
  var request_time = new Date();
  var request_method = tokens.method(req, res);
  var request_url = tokens.url(req, res);
  var status = tokens.status(req, res);
  var remote_addr = tokens['remote-addr'](req, res);
  if(req.session){
    var username = req.session['username'] || 'notlogin';
  }else {
    var username = 'notlogin';
  }

  // 直接将用户操作记入mysql中
  if(username != 'notlogin'){
    console.log(`方法: ${request_method}, URL: ${request_url}, 状态: ${status}, IP: ${remote_addr}, 用户名: ${username}`);
  }
}));
```

```

logDAO.userlog([username,request_time,request_method,request_url,status,remote
_addr], function (success) {
    console.log('成功保存! ');
})
}

```

logDAO.js

直接插入对应数据

```

userlog :function (useraction, callback) {
    pool.query('insert into
user_action(username,request_time,request_method,request_url,status,remote_addr
) values(?, ?,?,?,?,?,?)',
        useraction, function (error, result) {
            if (error) throw error;
            callback(result.affectedRows > 0);
        });
}

```

3.爬虫数据查询结果列表支持分页和排序

页面效果

序号	标题	作者	关键词	链接	发布时间
1	印度总理莫迪亲属因新冠肺炎离世	央视	莫迪,印度报业托拉斯,印度总理,新冠肺炎,全球多国爆发新冠肺炎疫情	https://news.sina.com.cn/w/2021-04-28/doc-ikmxzfmk9344740.shtml	2021-04-27T16:00:00.000Z
2	北京4月27日无新增新冠肺炎确诊 病例	新浪新闻综合	新冠肺炎,北京,新型冠状病毒肺炎疫情	https://news.sina.com.cn/c/2021-04-28/doc-ikmxzfmk9369508.shtml	2021-04-27T16:00:00.000Z
3	泰国新增新冠肺炎确诊病例2012例 累计确诊超6万例	央视	泰国,新冠肺炎,新型冠状病毒肺炎疫情	https://news.sina.com.cn/w/2021-04-28/doc-ikmxzfmk9410757.shtml	2021-04-27T16:00:00.000Z
4	香港新增新冠肺炎确诊病例7例 累 计确诊11755例	央视	新冠肺炎,香港,新型冠状病毒肺炎疫情	https://news.sina.com.cn/c/2021-04-28/doc-ikmyawc2332623.shtml	2021-04-27T16:00:00.000Z
5	瑞典新增新冠肺炎确诊病例7266例 累计确诊960520例	央视	新冠肺炎,全球多国爆发新冠肺炎疫情	https://news.sina.com.cn/w/2021-04-28/doc-ikmxzfmk9543572.shtml	2021-04-27T16:00:00.000Z

主页面前端代码

news.html

进行导航条设置以及相应的页面跳转

```

<html ng-app="news">
<head>
    <meta charset="utf-8">
    <title>News Detect</title>

    <link rel="stylesheet" href="https://cdn.staticfile.org/twitter-
bootstrap/3.3.7/css/bootstrap.min.css">
    <script src="https://cdn.staticfile.org/jquery/2.1.1/jquery.min.js">
</script>
    <script src="http://cdn.bootcss.com/bootstrap/3.3.4/js/bootstrap.min.js">
</script>
    <script type="text/javascript"
src="https://cdn.jsdelivr.net/npm/echarts@4.7.0/dist/echarts.min.js"></script>
    <script src='javascripts/dist/echarts-wordcloud.min.js'></script>
    <script src="/angular/angular.min.js"></script>

    <script src="javascripts/news.js" type="text/javascript"></script>

</head>
<body ng-controller="news_Ctrl" ng-init="isShow=false">
<nav class="navbar navbar-default navbar-fixed-top">
    <div class="container">
        <div class="navbar-header">
            <!-- <div class="nav" > -->
            <a class="navbar-brand" href="#">News DETECT</a>
        </div>
        <div id="navbar" class="navbar-collapse collapse">
            <!-- <div id="navbar" class="nav" > -->
            <ul class="nav navbar-nav">
                <!-- <ul class="nav" > -->
                <!-- <ul> -->
                <li ><a ng-click="showSearch()">检索</a></li>
                <li class="dropdown">
                    <a href="#" class="dropdown-toggle" data-toggle="dropdown">
可视化<span class="caret"></span></a>
                    <ul class="dropdown-menu">
                        <li><a ng-click="histogram()">柱状图</a></li>
                        <li><a ng-click="pie()">饼状图</a></li>
                        <li><a ng-click="line()">折线图</a></li>
                        <li><a ng-click="wordcloud()">词云</a></li>
                    </ul>
                </li>
                <li>
                    <a href="#" class="dropdown-toggle" data-toggle="dropdown"
ng-click="check()">后台管理<span class="caret"></span></a>
                    <ul class="dropdown-menu">
                        <!-- <li><a >资质审核</a></li> -->
                        <li><a ng-click="searchlog()">操作记录查看</a></li>

```

```

        <li><a href="http://localhost:3000/admin.html">用户管
理</a></li>
        </ul>

        </li>
        <li>
            <a href="#" class="dropdown-toggle" data-toggle="dropdown">
账号管理<span class="caret"></span></a>
            <ul class="dropdown-menu">
                <li class="dropdown-header">账号</li>
                <li><a ng-click="logout()">退出登录</a></li>
            </ul>
        </li>
    </ul>
</div>
</div>
</nav>
<!-- 所有的图片都绘制在main1位置-->
```

angularJS显示与隐藏实现细节

在文件中要依据不同实现效果修改不同元素 ng-show 或 ng-hide 指令表达式的指令

angularJS中的ng-show、ng-hide、ng-if指令都可以用来控制dom元素的显示或隐藏。ng-show和ng-hide根据所给表达式的值来显示或隐藏HTML元素。当赋值给ng-show指令的值为false时元素会被隐藏，值为true时元素会显示。ng-hide功能类似，使用方式相反。元素的显示或隐藏是通过改变CSS的display属性值来实现的。

```

<span ng-hide="isShow" ng-show="isPic" id="main1" style="width:
1000px;height:600px;position:fixed; top:70px;left:80px"></span>

<div ng-show="isShow" ng-show="isPic" style="width: 1300px;position:relative;
top:70px;left: 80px">
    <!-- 查询页面-->
    <div ng-include="'search.html'"></div>

</div>
<div ng-hide="isShow" style="width: 1300px;position:relative; top:70px;left:
80px">
    <!-- 查询log页面-->
    <div ng-include="'searchlog.html'"></div>
</div>
```

实现布尔表达式

1) 当点击“检索”按钮→调用 show_search() 函数

angular代码 (javascripts/new.js)

在 showSearch() 函数中，将 isShow 设为 true， isPic 设置为 false, ng-show 指令在表达式为 true 时显示指定的 HTML 元素，否则隐藏指定的 HTML 元素， ng-hide 相反。于是显示 isShow 表达式所在区域。设置选择框的默认项，如 selectTitle 默认为 and 等。

```
$scope.showSearch = function () {
    $scope.isShow = true;
    $scope.isisshowresult = false;
    $scope.isisshowlog = false;
    $scope.isPic=false;
    // 再次回到查询页面时，表单里要保证都空的
    $scope.title1=undefined;
    $scope.title2=undefined;
    $scope.selectTitle='AND';
    $scope.content1=undefined;
    $scope.content2=undefined;
    $scope.selectContent='AND';
    $scope.sorttime=undefined;
    $scope.sortscore=undefined;
    $scope.selectKeyword='AND';
    $scope.keyword1=undefined;
    $scope.keyword2=undefined;
    $scope.selectSource=undefined;
};
```

2) 显示输入框

search.html

news.html 中引入查询页面

```
<div ng-show="isShow" ng-init="isShowtext=false" style="width:1300px;position:relative; top:70px;left: 80px">
    <!--查询页面-->
    <div ng-include="'search.html'"></div>
```

调用 search 页面，其中搜索包括分词关键词搜索,标题, 内容, 关键词, 来源的相关筛选

```
<form class="form-horizontal" role="form">
    <div class="row" style="margin-bottom: 10px;">
        <label class="col-lg-2 control-label">分词关键字</label>
        <div class="col-lg-3">
            <input type="text" class="form-control" placeholder="请输入分词关键字" ng-model="$parent.splitword">
        </div>
```

```
<br></br>
<label class="col-lg-2 control-label">标题</label>
<div class="col-lg-3">
    <input type="text" class="form-control" placeholder="请输入标题关键字" ng-model="$parent.title1">
</div>
<div class="col-lg-1">
    <select class="form-control" autocomplete="off" ng-model="$parent.selectTitle">
        <option selected="selected">AND</option>
        <option>OR</option>
    </select>
</div>
<div class="col-lg-3">
    <input type="text" class="form-control" placeholder="请输入标题关键字" ng-model="$parent.title2">
</div>
</div>

<div class="row" style="margin-bottom: 10px;">
    <label class="col-lg-2 control-label">内容</label>
    <div class="col-lg-3">
        <input type="text" class="form-control" placeholder="请输入内容关键字" ng-model="$parent.content1">
    </div>
    <div class="col-lg-1">
        <select class="form-control" autocomplete="off" ng-model="$parent.selectContent">
            <option selected="selected">AND</option>
            <option>OR</option>
        </select>
    </div>
    <div class="col-lg-3">
        <input type="text" class="form-control" placeholder="请输入内容关键字" ng-model="$parent.content2">
    </div>
</div>

<div class="row" style="margin-bottom: 10px;">
    <label class="col-lg-2 control-label">关键词</label>
    <div class="col-lg-3">
        <input type="text" class="form-control" placeholder="请输入关键词" ng-model="$parent.keyword1">
    </div>
    <div class="col-lg-1">
        <select class="form-control" autocomplete="on" ng-model="$parent.selectKeyword">
            <option selected="selected">AND</option>
```

```

        <option>OR</option>
    </select>
</div>
<div class="col-lg-3">
    <input type="text" class="form-control" placeholder="请输入关键词"
ng-model="$parent.keyword2">
</div>
</div>
<div class="row" style="margin-bottom: 10px;">
    <label class="col-lg-2 control-label">来源</label>
    <div class="col-lg-1">
        <select class="form-control" autocomplete="off" ng-
model="$parent.selectSource">
            <option selected="selected">新浪新闻</option>
            <option>人民网</option>
            <option>网易新闻</option>
        </select>
    </div>
</div>

<div class="form-group">
    <div class="col-md-offset-9">
        <button type="submit" class="btn btn-default" ng-click="search()">
查询</button>
    </div>
</div>

</form>

```

查询时间排序

```

<button type="submit" class="btn btn-primary" ng-
click="searchsortASC()" >发布时间升序</button>
<button type="submit" class="btn btn-primary" ng-
click="searchsortDESC()">发布时间降序</button>

```

3) 调用 search() 函数

javascripts/news.js

获取用户传入的文本内容，先检查参数是否存在问题，再通过GET请求传给路由。

如果获取到了数据，则显示表格查询结果并进行分页；

如果没有获取到用户登录信息，返回登录页面。

```

$scope.search = function () {

```

```

var splitword=$scope.splitword;
var title1 = $scope.title1;
var title2 = $scope.title2;
var selectTitle = $scope.selectTitle;
var content1 = $scope.content1;
var content2 = $scope.content2;
var selectContent = $scope.selectContent;
var keyword1 = $scope.keyword1;
var keyword2 = $scope.keyword2;
var selectKeyword = $scope.selectKeyword;
var sorttime = $scope.sorttime;
var sortscore = $scope.sortscore;
var selectSource=$scope.selectSource;

// 检查用户传的参数是否有问题
//用户有可能这样输入: ___ and/or 新冠（直接把查询词输在了第二个位置）
if(typeof title1=="undefined" && typeof title2!="undefined" &&
title2.length>0){
    title1 = title2;
}
if(typeof content1=="undefined" && typeof content2!="undefined" &&
content2.length>0){
    content1 = content2;
}
if(typeof keyword1=="undefined" && typeof keyword2!="undefined" &&
keyword2.length>0){
    keyword1 = keyword2;
}
// 用户可能一个查询词都不输入， 默认就是查找全部数据
var myurl = `/news/search?
t1=${title1}&ts=${selectTitle}&t2=${title2}&c1=${content1}&cs=${selectContent}&
c2=${content2}&k1=${keyword1}&ks=${selectKeyword}&k2=${keyword2}&ss=${selectSou
rce}&stime=${sorttime}&sscore=${sortscore}&t=${splitword}`;

$http.get(myurl).then(
    function (res) {
        if(res.data.message=='data'){
            $scope.isisshowresult = true; //显示表格查询结果
            // $scope.searchdata = res.data;
            $scope.initPageSort(res.data.result)
        }else {
            window.location.href=res.data.result;
        }
    }

),function (err) {
    $scope.msg = err.data;
});
};

```

initPageSort函数

```
$scope.initPageSort=function(item){  
    $scope.pageSize=5;      //每页显示的数据量，可以随意更改  
    $scope.selPage = 1;  
    $scope.data = item;  
    $scope.pages = Math.ceil($scope.data.length / $scope.pageSize); //分页数  
    $scope.pageList = [];//最多显示5页，后面6页之后不会全部列出页码来  
    $scope.index = 1;  
  
    var len = $scope.pages> 5 ? 5:$scope.pages;  
    $scope.pageList = Array.from({length: len}, (x,i) => i+1);  
  
    //设置表格数据源(分页)  
    $scope.items = $scope.data.slice(0, $scope.pageSize);  
  
};
```

查询时间排序

```
$scope.searchsortASC = function () {  
    $scope.sorttime = '1';  
    $scope.search();  
};  
$scope.searchsortDESC = function () {  
    $scope.sorttime = '2';  
    $scope.search();  
};
```

4) 传给路由

routes/news.js

这里判断用户是否登陆已经在第一个功能非注册用户不能查看中有所涉及，这里不做赘述

```

router.get('/search', function(request, response) {
    console.log(request.session['username']);
    //sql字符串和参数
    if (request.session['username'] === undefined) {
        // response.redirect('/index.html')
        response.json({message: 'url', result: '/index.html'});
    }else {
        var param = request.query;
        newsDAO.search(param, function (err, result, fields) {
            response.json({message: 'data', result: result});
        })
    }
});

```

在连接池文件 newsDao.js 中对关键词和连接词执行拼路由操作，将查询结果按照新闻发表时间排序

这里and where判断有所讲究，判断是否前面的关键词查询中都无输入才进行这次的查询。包含标题，内容，关键词，来源，查询时间的sql语句构造。

```

if(searchparam["t2"]!="undefined"){
    sql +=(`where title like '%${searchparam["t1"]}%'
${searchparam['ts']} title like '%${searchparam["t2"]}%' `);
} else if(searchparam["t1"]!="undefined"){
    sql +=(`where title like '%${searchparam["t1"]}%' `);
};

if(searchparam["t1"]=="undefined" && searchparam["t2"]=="undefined" && searchparam["c1"]!="undefined"){
    sql+='where ';
} else
if(searchparam["t1"]!="undefined" && searchparam["c1"]!="undefined"){
    sql+='and ';
}

if(searchparam["c2"]!="undefined"){
    sql +=(`content like '%${searchparam["c1"]}% ${searchparam['cs']}
content like '%${searchparam["c2"]}%' `);
} else if(searchparam["c1"]!="undefined"){
    sql +=(`content like '%${searchparam["c1"]}%' `);
}

if(searchparam["c1"]=="undefined" && searchparam["c2"]=="undefined" && searchparam["t1"]=="undefined" && searchparam["t2"]=="undefined" && searchparam["k1"]!="undefined"){
    sql+='where ';
}

```

```

        }else
if((searchparam[ "t1" ]!="undefined" || searchparam[ "c1" ]!="undefined")&&searchparam[ "k1" ]!="undefined"){
    sql+='and ';
}

if(searchparam[ "k2" ]!="undefined"){
    sql +=(`keywords like '%${searchparam[ "k1" ]}% ${searchparam[ 'ks' ]}%
keywords like '%${searchparam[ "k2" ]}%` );
} else if(searchparam[ "k1" ]!="undefined"){
    sql +=(`keywords like '%${searchparam[ "k1" ]}%` );
}
console.log(searchparam[ "ss" ]);

if(searchparam[ "c1" ]=="undefined" &&searchparam[ "c2" ]=="undefined" &&searchparam[ "t1" ]=="undefined" &&searchparam[ "t2" ]=="undefined" &&searchparam[ "k1" ]=="undefined" &&searchparam[ "k2" ]=='undefined' &&searchparam[ "ss" ]!='undefined'){
    sql+='where ';
    console.log("yes!")
} else
if((searchparam[ "t1" ]!="undefined" || searchparam[ "c1" ]!="undefined" || searchparam[ "k1" ]!="undefined" )&&searchparam[ "ss" ]!="undefined"){
    sql+='and ';
}
if(searchparam[ "ss" ]=="undefined"){
    // sql +=(`*****`);
    console.log("*****")
}
if(searchparam[ "ss" ]!="undefined"){
    console.log("source")
    sql +=(`source_name = '${searchparam[ "ss" ]}'` );
    console.log(sql)
}

if(searchparam[ 'stime' ]!="undefined"){
    if(searchparam[ 'stime' ]=="1"){
        sql+='ORDER BY publish_date ASC ';
    }else {
        sql+='ORDER BY publish_date DESC ';
    }
}
sql+='';

console.log(searchparam[ "ks" ]);
console.log(sql)
pool.getConnection(function(err, conn) {
    if (err) {
        callback(err, null, null);
    } else {
        conn.query(sql, function(qerr, vals, fields) {

```

```

        conn.release(); //释放连接
        callback(qerr, vals, fields); //事件驱动回调
    });
}
);

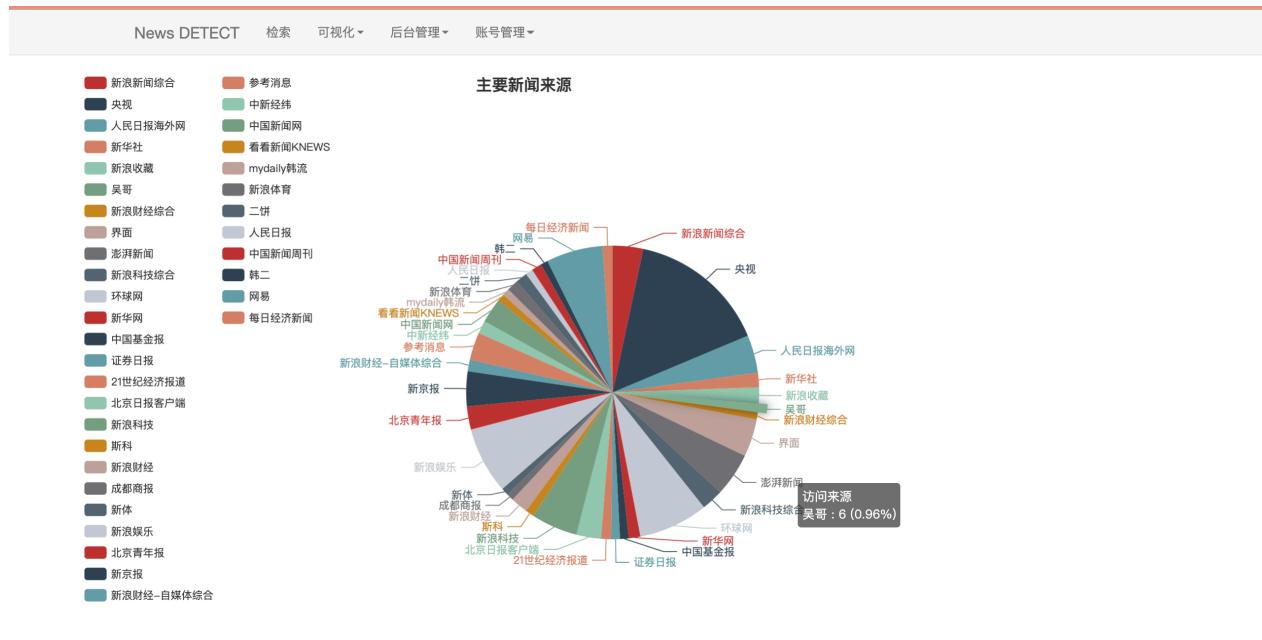
```

4.用Echarts或者D3实现3个以上的数据分析图表展示在网站中 以饼图为例

1) 点击饼图—>调用 showsearchline() 函数

由于人民网的作者结果为数字，所以要进行正则表达式匹配筛选

实现效果



```

$scope.pie = function () {
    $scope.isShow = false;
    $scope.isPic=true;
    $http.get("/news/pie").then(
        function (res) {
            if(res.data.message=='url'){
                window.location.href=res.data.result;
            }else {
                let newdata = [];
                res.data.result.forEach(function (element) {
                    console.log(element["x"].match)

                    //不包含数字
                    var reg = /^\\d+$/;

```

```
//不为空
var reg1= /\S/;
console.log(reg1.test(element["x"]))
if(!reg.test(element["x"])&&reg1.test(element["x"])){
    newdata.push({name: element["x"], value:
element["y"]});
}

});

var myChart =
echarts.init(document.getElementById('main1'));
var app = {};
option = null;
// 指定图表的配置项和数据
var option = {
    title: {
        text: '主要新闻来源',
        x: 'center'
    },
    tooltip: {
        trigger: 'item',
        formatter: "{a} <br/>{b} : {c} ({d}%)"
    },
    legend: {
        orient: 'vertical',
        left: 'left',
        // data: ['直接访问', '邮件营销', '联盟广告', '视频广
告', '搜索引擎']
    },
    series: [
        {
            name: '访问来源',
            type: 'pie',
            radius: '55%',
            center: ['60%', '60%'],
            data: newdata,
            itemStyle: {
                emphasis: {
                    shadowBlur: 10,
                    shadowOffsetX: 0,
                    shadowColor: 'rgba(0, 0, 0, 0.5)'
                }
            }
        }
    ]
};
// myChart.setOption(option);
app.currentIndex = -1;
```

```

        setInterval(function () {
            var dataLen = option.series[0].data.length;
            // 取消之前高亮的图形
            myChart.dispatchAction({
                type: 'downplay',
                seriesIndex: 0,
                dataIndex: app.currentIndex
            });
            app.currentIndex = (app.currentIndex + 1) % dataLen;
            // 高亮当前图形
            myChart.dispatchAction({
                type: 'highlight',
                seriesIndex: 0,
                dataIndex: app.currentIndex
            });
            // 显示 tooltip
            myChart.dispatchAction({
                type: 'showTip',
                seriesIndex: 0,
                dataIndex: app.currentIndex
            });
        }, 1000);
        if (option && typeof option === "object") {
            myChart.setOption(option, true);
        }
    ;
}
);
}
;

```

2.前端

点击饼状图

```

<ul class="dropdown-menu">
    <li><a ng-click="histogram()">柱状图</a></li>
    <li><a ng-click="pie()">饼状图</a></li>
    <li><a ng-click="line()">折线图</a></li>
    <li><a ng-click="wordcloud()">词云</a></li>
</ul>

```

3.传给路由

routes/news.js

```

router.get('/pie', function(request, response) {
    //sql字符串和参数
    console.log(request.session['username']);
}

```

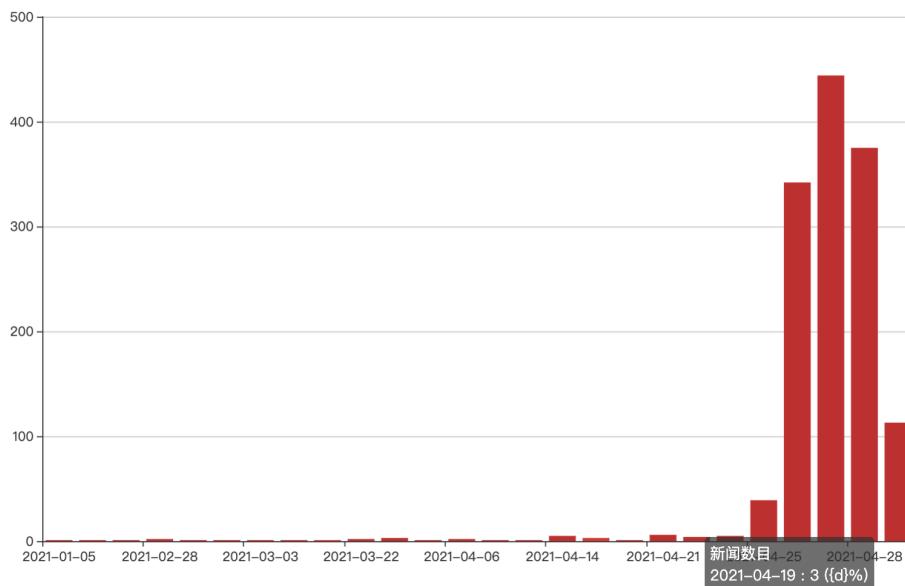
```
//sql字符串和参数
if (request.session['username']==undefined) {
    // response.redirect('/index.html')
    response.json({message:'url',result:'/index.html'});
} else {
    var fetchSql = "select author as x,count(author) as y from fetches
group by author having y >=5;";
    newsDAO.query_noparam(fetchSql, function (err, result, fields) {
        response.writeHead(200, {
            "Content-Type": "application/json",
            "Cache-Control": "no-cache, no-store, must-revalidate",
            "Pragma": "no-cache",
            "Expires": 0
        });
        response.write(JSON.stringify({message:'data',result:result}));
        response.end();
    });
}
});
```

词云图



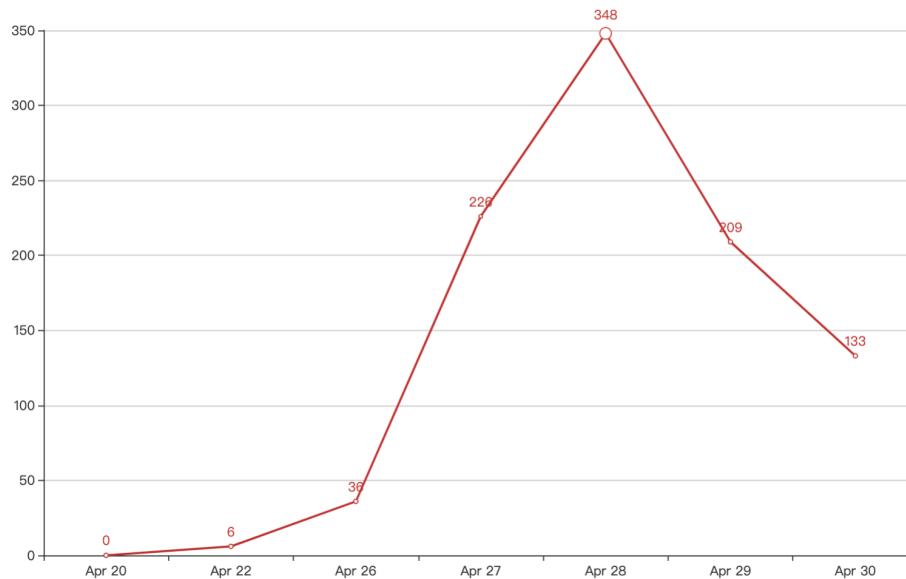
柱状图

新闻发布数 随时间变化



折线图

"疫情"该词在新闻中的出现次数随时间变化图



5. 实现一个管理端界面，可以查看（查看用户的操作记录）和管理（停用启用）注册用户。

资质审核

前端点按时调用check函数

```
<a href="#" class="dropdown-toggle" data-toggle="dropdown" ng-click="check()">
后台管理<span class="caret"></span></a>
```

调用 check() 函数

news.js

判断是否登陆以及是否为管理员并进行相应的跳转操作

```
$scope.check= function () {
    var data = JSON.stringify({
        username: $scope.username
    });
    $http.get( "/users/check" , data)
        .then(
            function (res) {
                console.log(data)
                console.log("msg",res.data.msg)
                if(res.data.msg=='url'){
                    window.alert("请先登陆")
                    window.location.href='/index.html';
                }
                else if(res.data.msg=='欢迎您，管理员wwq') {
                    window.alert("欢迎您，管理员wwq")
                }
                else{
                    $scope.msg=res.data.msg;
                    window.alert($scope.msg);
                }
            },
            function (err) {
                $scope.msg = err.data;
                window.alert($scope.msg);
            })
        );
};
```

路由代码

users.js

提示语包括如果username为undefined跳转登陆界面

判断管理员身份成功，显示'欢迎您，管理员wwq'

否则显示'对不起，您不是管理员，您无法访问后台管理界面"

```
router.get( '/check' , function(req, res,next) {
    console.log(req.session['username']);
    //sql字符串和参数
    if (req.session[ 'username' ]==undefined) {
```

```

        // response.redirect('/index.html')
        res.json({msg:'url',result:'/index.html'});
    }
    else if (req.session['username']=='wwq') {
        res.json({msg:"欢迎您, 管理员wwq"});
        // response.redirect('/index.html')
    }else {
        res.json({msg:"对不起, 您不是管理员, 您无法访问后台管理界面"})
    }
}
);

```

查看操作日志

前端

```

<ul class="dropdown-menu">
    <!-- <li><a >资质审核</a></li> -->
    <li><a ng-click="searchlog()">操作记录查看</a></li>
    <li><a href="http://localhost:3000/admin.html">用户管
理</a></li>
</ul>

```

点按操作记录查看

调用searchlog函数

以表格形式显示, 支持分页

```

$scope.searchlog = function () {

    // $http.get().then();
    $scope.isShow = false;
    $scope.isPic = false;
    $http.get("/news/searchlog").then(
        function (res) {
            if(res.data.message=='data'){
                $scope.isisshowlog = true; //显示表格查询结果
                // $scope.searchdata = res.data;
                console.log(res.data.result)
                $scope.initPageSort(res.data.result)
            }else {
                window.location.href=res.data.result;
            }
        },
        function (err) {
            $scope.msg = err.data;
        }
    );
}

```

```
    });
};

});
```

路由代码

news.js

```
router.get('/searchlog', function(request, response) {
  console.log(request.session['username']);
  //sql字符串和参数
  if (request.session['username'] === undefined) {
    // response.redirect('/index.html')
    response.json({message: 'url', result: '/index.html'});
  } else {
    // var param = request.query;
    logDAO.searchlog(function (err, result, fields) {
      response.json({message: 'data', result: result});
    })
  }
});
```

logDAO.js

mysql查询语句select * from user_action group by id ;

```
searchlog :function( callback) {
  // 组合查询条件
  var sql = 'select * from user_action group by id';
  console.log(sql)
  pool.getConnection(function(err, conn) {
    if (err) {
      callback(err, null, null);
    } else {
      conn.query(sql, function(qerr, vals, fields) {
        conn.release(); //释放连接
        callback(qerr, vals, fields); //事件驱动回调
      });
    }
  });
},
```

停用启用

停用启用用户操作在第一个功能中已经描述过具体的实现方式，这里只需点击用户管理，即可进入停用启用用户界面

```
<a href="#" class="dropdown-toggle" data-toggle="dropdown" ng-click="check()">
后台管理<span class="caret"></span></a>
    <ul class="dropdown-menu">
        <!-- <li><a >资质审核</a></li> -->
        <li><a ng-click="searchlog()">操作记录查看</a></li>
        <li><a href="http://localhost:3000/admin.html">用户管
理</a></li>
    </ul>
```

6. 实现对爬虫数据中文分词的查询

这里使用基于 IKAnalyzer 字典分词器的 node.js 实现——node-analyzer

添加分词框

News DETECT 检索 可视化后台管理 账号管理

分词关键字

标题 AND

内容 AND

关键词 AND

来源

分词处理

首先建立表格存储分词结果

在createtable.sql中添加

```
CREATE TABLE WordSplit (
    id_fetches int,
    word varchar(50) DEFAULT NULL
);
```

在javasripts/stopwords.js 进行分词处理

对爬虫内容，先用正则表达式去掉一些无用的字符；进行分词后，逐个处理词条判断是否存在于停用词表中（停用词表用的是哈工大停用词表），将经过筛选后的词条存储到数据库的 WordSplit 表中。

```
const regex = /[\t\s\r\n\d\w]+[+\-\(\),\.\.,\!\?\{\}\[\]\":\%\-\/\"]+/g;
var fetchSql = "select id_fetches,content from fetches;";
console.log("enter stopword.js")
mysql.query_noparam(fetchSql, function (err, result, fields) {
    result.forEach(function (item){
        var segmenter = new Segmenter();
```

```
var newcontent = item["content"].replace(regex,'');
if(newcontent.length !== 0){
    var words = segmenter.analyze(newcontent).split(' ');
    var id_fetch = item["id_fetches"];
    words.forEach(function(word){
        if(!stop_words.has(word)&&word.length>1){
            var wordSql = "insert into WordSplit (id_fetches,word)
values('" + id_fetch + "','" + word+"');";
            console.log(wordSql);
            mysql.query(wordSql, function(err, result){
                console.log("enter query")
                if(err)
                    {console.log(err);
                }
            });
        }
    });
}
});
```

存储成功

```
[mysql] select * from WordSplit limit 10;
+-----+-----+
| id_fetches | word      |
+-----+-----+
|       1 | 标题      |
|       1 | 开放      |
|       1 | 南宁      |
|       1 | 来源      |
|       1 | 广西      |
|       1 | 网络      |
|       1 | 广播 电视 |
|       1 | 责任 编辑 |
|       2 | 标题      |
|       2 | 一季度    |
+-----+-----+
10 rows in set (0.05 sec)
```

分词查询

前端除了添加分词输入框外还要将相应的值进行传递

```

var myurl = `/news/search?
t1=${title1}&ts=${selectTitle}&t2=${title2}&c1=${content1}&cs=${selectContent}&
c2=${content2}&k1=${keyword1}&ks=${selectKeyword}&k2=${keyword2}&ss=${selectSou
rce}&stime=${sorttime}&sscore=${sortscore}&t=${splitword}`;

```

newsDAO.js

在此之前一开始初始化sql语句时还需判断是否用户键入了分词关键词，并在from后添加对应的表格WordSplit

对用户提交的内容也要进行分词

如北京的天气 中间的“的”就是一个停用词，没有实际意义，也没有储存在之前的分词结果中

若用户提交的字符串较短时，直接到数据库索引词汇；

否则，先进行分词操作，再对每个词条进行判断，这里把在停用词表中的词或者长度为一的词条当作停用词，需要将它们剔除，然后在 WordSplit 表中获取每个分词的 id 字段，最后取 id 字段交集查询 fetches 表中完整的新闻爬取信息。

具体的最后效果以及在之前search部分以及demo中予以演示。

```

if(searchparam["t"]!="undefined"){
    if(searchparam["t"].length<=3){
        sql+=(`where id_fetches in (select id_fetches from WordSplit
where word like '${searchparam["t"]}');

    }else{
        var newcontent = searchparam["t"].replace(regex,'');
        var words = segmenter.analyze(newcontent).split(' ');
        var n=1;
        sql+=(`where id_fetches in (select id_fetches from(select
id_fetches from WordSplit where word like '${words[0]}')`);
        for(var i=1;i<words.length;i++){
            if(!stop_words.has(words[i])&&words[i].length>1){
                sql+=(` UNION ALL select id_fetches from WordSplit
where word like '${words[i]}');

                n++;
            }
        }
        sql+=(`)a GROUP BY id_fetches HAVING COUNT(*) = ${n})`;
    }
}

```

7. 实现查询结果按照主题词打分的排序

打分机制 (TF-IDF)

词频 (term frequency, TF) 指的是某一个给定的词语在该文件中出现的次数。这个数字通常会被归一化(一般是词频除以文章总词数)，以防止它偏向长的文件。

逆向文件频率 (inverse document frequency, IDF) IDF的主要思想是：如果包含词条t的文档越少，IDF越大，则说明词条具有很好的类别区分能力。某一特定词语的IDF，可以由总文件数目除以包含该词语之文件的数目，再将得到的商取对数得到。

$$TF - IDF = TF * IDF$$

在Javascript/rank.js中实现

需要首先建立sql表格

```
CREATE TABLE Score (
    id_fetches int,
    word varchar(50) DEFAULT NULL,
    weight float
);
```

计算TF

对应id文档中word出现次数：

```
select count(*) as num from WordSplit where word='${word}' and
id_fetches=${id};
```

对应id文档中词条总数目

```
select count(*) as num from WordSplit where id_fetches=${id};
```

计算IDF

文档总数

```
select count(distinct id_fetches) as num from WordSplit;
```

包含word的文档数

```
select count(distinct id_fetches) as num from WordSplit where word='${word}';
```

写入Score表中

```
var insert_word_Sql = 'INSERT INTO Score(id_fetches,word,weight)
VALUES(?, ?, ?);';
var insert_word_Params = [id,word,weight];
newsDAO.query(insert_word_Sql, insert_word_Params,
function(err){
```

带参数的query

newsDAO.js中添加

```
query :function(sql,param, callback) {
    pool.getConnection(function(err, conn) {
        if (err) {
            callback(err, null, null);
        } else {
            conn.query(sql,param, function(qerr, vals, fields) {
                console.log(sql);
                console.log(param);
                conn.release(); //释放连接
                callback(qerr, vals, fields); //事件驱动回调
            });
        }
    });
},
```

打分排序存储结果

```
[mysql] > select * from Score limit 30;
+-----+-----+-----+
| id_fetches | word | weight |
+-----+-----+-----+
| 2 | 排行 | 0.103693 |
| 1 | 网络 | 0.213094 |
| 2 | 职业 | 0.155539 |
| 1 | 南宁 | 0.299737 |
| 2 | 看看 | 0.0259232 |
| 1 | 开放 | 0.213094 |
| 1 | 标题 | 0.0250838 |
| 2 | 全国 | 0.0140463 |
| 2 | 标题 | 0.00108471 |
| 1 | 广西 | 0.16241 |
| 2 | 看看 | 0.0259232 |
| 2 | 职业 | 0.155539 |
| 2 | 出炉 | 0.0129616 |
| 2 | 第一季度 | 0.0259232 |
| 2 | 全国 | 0.0140463 |
| 1 | 广播电视 | 0.299737 |
| 1 | 责任编辑 | 0.0250838 |
| 2 | 城市 | 0.00921485 |
| 2 | 快来 | 0.0129616 |
| 2 | 大于求 | 0.0259232 |
| 2 | 一季度 | 0.0259232 |
| 2 | 大于求 | 0.0259232 |
| 2 | 中意 | 0.0129616 |
| 2 | 招聘 | 0.0518464 |
| 2 | 招聘 | 0.0518464 |
| 2 | 第四季度 | 0.0129616 |
| 2 | 特点 | 0.00921485 |
| 2 | 相比 | 0.0129616 |
| 2 | 排行 | 0.103693 |
| 2 | 排行 | 0.103693 |
+-----+-----+-----+
30 rows in set (0.01 sec)
```

后端查询语句

其余实现类似基于时间的排序，这里考虑到一般都是搜打分从高到低，所以是实现了打分降序排序，效果展示在demo中已经呈现

```
if(searchparam['sscore']=="undefined"){
    var sql = 'select * from fetches ';
} else{
    var sql ='select * from fetches,Score ';
}
```

```
if(searchparam['sscore']!="undefined"){
    sql+='fetches.id_fetches=Score.id_fetches '
    if(searchparam['stime']=="undefined"){
        sql+='ORDER BY weight DESC ';
    }
    else{
        sql+=',weight DESC ';
    }
}
```

相关错误和问题处理

1.js报404

问题原因：路径写错

路径写错是一种常见的问题，也是需要首先第一检查的，这里我的js路径和html不在一个路径下，使用相对路径会出现找不到文件的情况

2.Unknown column 'xx' in 'where clause'

拼凑sql语句时对字符类型数据没有用引号引起

更改为以下语句即可改正

3.cannot read property ** of undefined

```
tserv.js:457
    throw err; // Rethrow non-MySQL errors
^

TypeError: Cannot read property 'passwd' of undefined
    at Query.<anonymous> (/Users/wangwenqing/Desktop/web编程/10185501405 王文清 期中作业/代码/search_site/routes/index.js:25:23)
    at Query.<anonymous> (/Users/wangwenqing/Desktop/web编程/10185501405 王文清 期中作业/代码/search_site/node_modules/mysql/lib/Connection.js:526:10)
    at Query._callback (/Users/wangwenqing/Desktop/web编程/10185501405 王文清 期中作业/代码/search_site/node_modules/mysql/lib/Connection.js:488:16)
    at Query.Sequence.end (/Users/wangwenqing/Desktop/web编程/10185501405 王文清 期中作业/代码/search_site/node_modules/mysql/lib/protocol/sequences/Sequence.js:83:24)
    at Query._handleFinalResultPacket (/Users/wangwenqing/Desktop/web编程/10185501405 王文清 期中作业/代码/search_site/node_modules/mysql/lib/protocol/sequences/Query.js:149:8)
    at Query.EofPacket (/Users/wangwenqing/Desktop/web编程/10185501405 王文清 期中作业/代码/search_site/node_modules/mysql/lib/protocol/sequences/Query.js:133:8)
    at Protocol._parsePacket (/Users/wangwenqing/Desktop/web编程/10185501405 王文清 期中作业/代码/search_site/node
```

需要添加判断

```
var select_Sql1 = "select * from UserInfo where name ='" + username + "'";  
let value1 = await mysql.promise_query(select_Sql1, function () {});
```

如上代码获取的value1的length是否为0，如果为0，需要进行特殊处理，否则就会报这个错

4.Truncated incorrect DOUBLE value

原因:字符串要加引号,即使是数值。

5. Cannot set headers after they are sent to the client

原因：客户端发出一次请求，服务器给出两次及以上响应

解决方案：清理多余次数响应；每次响应后立马return掉函数

6. 出现mysql查询正常但是无法写入的情况

需要给root用户授权从客户端访问

```
alter user 'root'@'localhost' identified with mysql_native_password by 'root';
flush privileges;
```

即可解决

7. 分词效果不理想

手动配置停用词

没有手动配置前的结果



手动配置后的结果

手动配置

```
const regex = /[\t\s\r\n\d\w|[\+\-\(\),\.\.,!\?()；你我他她－把及的是和又在了#\《》@\[\]\":%\-\"/"\"]]/g;
```



总结

在本次实验中学习了运用的angular.js框架，该框架简化了代码的编写，进行了清晰的分层，提高了代码的可维护性。同时也进行了依照期中项目时直接与mysql交互的实现，并进行async和await处理。最终达到用户与应用程序交互时动态更新页面的效果，更好地响应用户操作的业务逻辑管理。对于可视化展示，进行正则表达式处理，帮助图像的清晰呈现。实现了扩展功能中的基于中文分词的查询以及TF-IDF打分和排序功能。同时也摸索出了较大型的sql语句的实现，如何利用前端传递的值进行相应的数据库交互。