### express+mysql登录注册功能

1. 安装express,和express生成器

npm install express -g

npm install express-generator -g

1. 创建工程目录，了解目录结构

express -e myappDemo01

1. 引入mysql

在package.json的dependencies中加上"mysql" : "latest"

1. 在app.js中引入解析post请求的中间件并使用

var bodyParser = require('body-parser');

app.use(bodyParser.urlencoded({ extended: false }));

1. 编写登录路由和注册路由，并在app.js引入

登录路由

router.post('/', (*req*, *res*, *callback*) => {

*console*.log(req.body);

    userDao.login(req.body.username, req.body.password, *result* => {

*console*.log(result);

        if(result.type == 'success') {

            res.render('main', { username: req.body.username });

        }else {

*console*.log(111);

            res.send(`<script>alert('${result.msg}');location.href='/'</script>`);

        }

    })

})

注册路由

var express = require('express');

var router = express.Router();

var userDao = require('../dao/userDao');

router.post('/', (*req*, *res*, *callback*) => {

*console*.log(req.body.username, req.body.password);

    userDao.regist(req.body.username, req.body.password, *result* => {

        // res.json({status: 200, result});

        res.send(`<script>alert('${result.msg}');location.href='/'</script>`);

    })

})

*module*.*exports* = router;

1. 配置Mysql连接

var mysql = require('mysql');

var connection = {};

connection.createConnection = mysql.createConnection({

    host: '192.168.6.170',

    user: 'root',

    password: 'root',

    database: 'test\_zzj',

    prot: '3306'

})

*exports*.getConnection = connection.createConnection;

1. 编写 处理后台数据的js

var login = require('./login');

var regist = require('./regist');

var loginDo = function(*username*, *password*, *callback*) {

    login.login(username, password, *res* => {

        callback(res);

    })

}

var registDo = function(*username*, *password*, *callback*) {

    regist.regist(username, password, *res* => {

        callback(res);

    })

}

*exports*.login = loginDo;

*exports*.regist = registDo;

登录

var connection = require('../configs/connection');

var login = function(*username*, *password*, *callback*) {

    let data = {};

    let sql = `select \* from user\_info where username = '${username}';` //注意给参数添加单引号

    connection.getConnection.query(sql, (*err*, *rows*, *fields*) => {

        if(err || !rows[0]) {

            data.type = 'fail';

            data.msg = '用户名不存在';

            callback(data);

        }else {

            if(rows[0].password == password) {

                data.type = 'success';

                data.msg = '登录成功';

                callback(data);

            }else {

                data.type = 'fail';

                data.msg = '密码错误';

                callback(data);

            }

        }

    })

}

*exports*.login = login

注册

var connection = require('../configs/connection');

var regist = function(*username*, *password*, *callback*) {

    let data = {};

    let sql = `insert into user\_info(username,password) values('${username}','${password}');`;

    connection.getConnection.query(sql, (*err*, *rows*, *field*) => {

        if(rows != undefined) {

            data.type = 'success';

            data.msg = '注册成功';

            callback(data);

        }else {

            data.type = 'fail';

            data.msg = '信息填写有误';

            callback(data);

        }

    })

}

*exports*.regist = regist;

根据路由跳转到不同界面

首页

 <h1>欢迎</h1>

    <div>

      <form *action*="/login" *method*="POST">

        userName: <input *type*="text" *name*="username" /> <br />

        password: <input *type*="password" *name*="password" /> <br />

        <input *type*="submit" *value*="登录">

        <input *type*="button" *value*="注册"  *onclick*="window.location.href='/goregist'">

    </form>

注册界面

 <h1>请注册</h1>

    <div>

      <form *action*="/regist" *method*="POST">

        userName: <input *type*="text" *name*="username" /> <br />

        password: <input *type*="password" *name*="password" /> <br />

        <input *type*="submit" *value*="注册" >

    </form>

登录成功界面

<body>

    你好*<*%= username %>

</body>

### 二、配置taf服务

利用之前做好的前端服务和后端taf服务，进行整合。

1. 在前端taf服务，编写一个jce接口文件
2. 通过up jce2node --client Hello.jce 生成一个代理js
3. 编写service.js 作为连接后端taf服务的桥梁

const TAF = require('@taf/taf-rpc').Communicator.New();

const Test = require('./HelloProxy').Test;

// 服务的标识名

let servant = 'Test.HelloZzjServer.HelloObj';

if (!process.env.TAF\_CONFIG) {

  // 后台服务的端口和ip

  servant += '@tcp -h 172.16.8.63 -t 60000 -p 10082';

}

// 客户端通过Taf.stringToProxy建立连接

const prx = TAF.stringToProxy(Test.HelloProxy, servant);

*exports*.Test = Test;

*exports*.prx = prx;

1. 在app路由里，引入service导出的Test 和 prx

app.get('/hello', (*req*, *res*) => {

    // 创建一个入参对象

    let stReq = **new** *service*.*Test*.*HelloWorldReq*();

    // 通过readFromObject 传入参数，并转成对象

    stReq.readFromObject({

        data: req.query.data

    });

*console*.log(stReq);

    // 通过代理调用接口并传入入参，根据返回结果进行处理

    service.prx.helloWorld(stReq).then(*ret* => {

        // 获取返回的出参

        res.json(ret.response.arguments.stRsp.toObject())

    }, *ret* => {

*console*.log(ret);

        res.json(res.response)

    })

1. 编写界面，请求对应的路由并传入参数

 <form *action*="/hello" *method*="GET">

        data: <input *type*="text" *name*="data" /> <br />

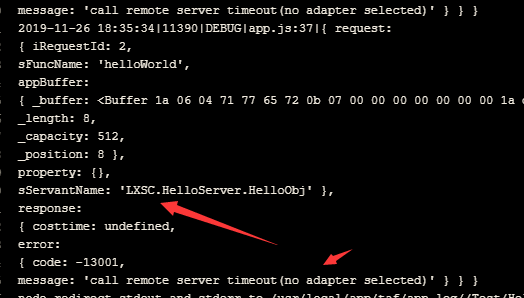
        <input *type*="submit" *value*="发送">

    </form>

1. 错误

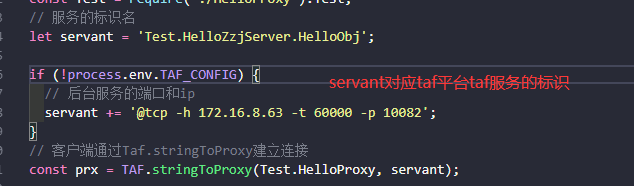
1.在jce文件中module名称没有和在taf平台配置的应用名对应上

导致web服务在平台上找不到对应应用而请求超时





1. 在连接后台taf服务的配置文件 service.js中 servant没有对应上服务名，导致找不到对应引用而请求超时



1. 将修改好的服务打包发布到平台，并配置域名进行访问

