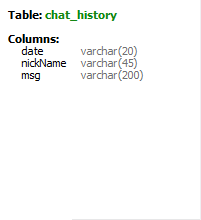
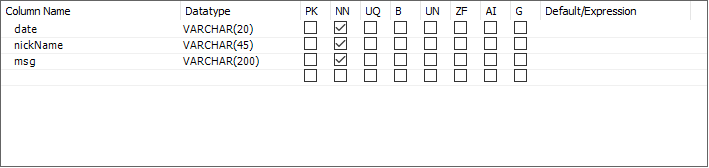
聊天信息数据规范

Mysql

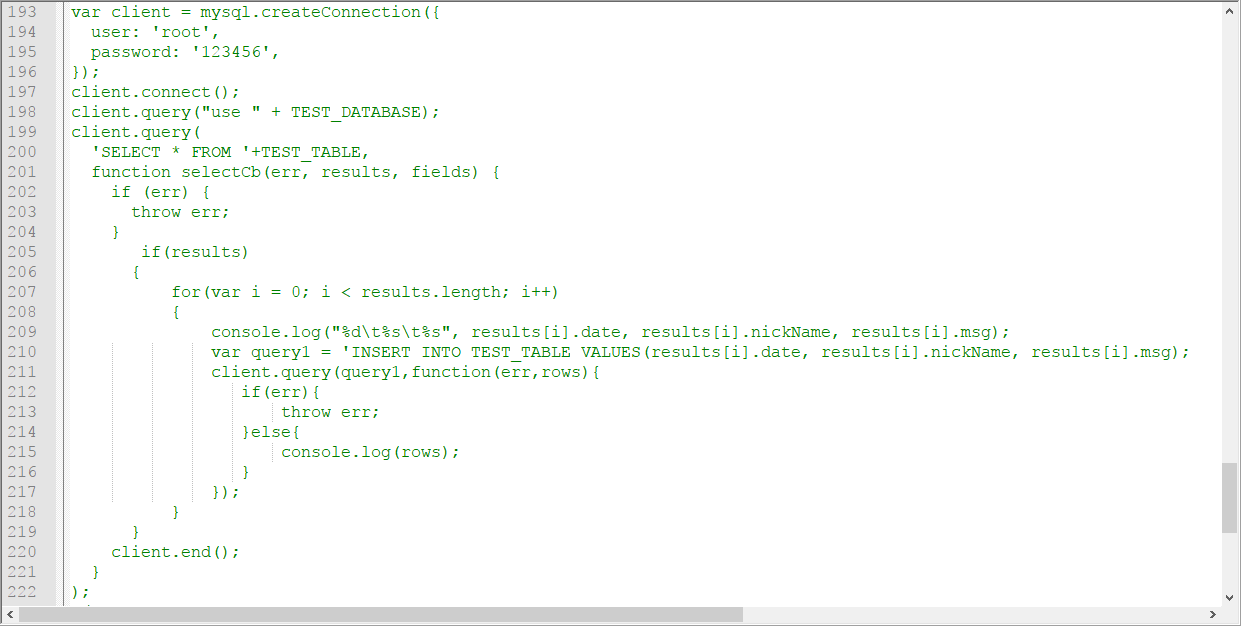


index.html



server.js

测试数据库为my\_news\_test库下的chat\_history表，字段为nickName、date、msg



**[javascript]** [view plain](http://blog.csdn.net/kongjunchao159/article/details/50232741" \o "view plain) [copy](http://blog.csdn.net/kongjunchao159/article/details/50232741" \o "copy)

1. **var** http = require('http');
2. **var** querystring = require('querystring');
3. **var** mysql = require('mysql');
5. //服务器端接收数据
6. **var** server = http.createServer(**function**(req,res){
7. **if**(req.url !== 'favicon.ico'){
8. **var** params;
9. req.on('data',**function**(data){
10. //使用querystring模块中的parse方法将字符串转化为对象
11. params = querystring.parse(decodeURIComponent(data));
12. })
13. req.on('end',**function**(){
14. console.log('客户端请求数据已全部接收完毕');
15. connect(params);
16. })
17. //使用Access-Control-Allow-Origin解决跨域问题
18. res.setHeader('Access-Control-Allow-Origin','\*');
19. //返回JSON数据
20. res.writeHead(200,{'Content-Type' : 'application/json'});
21. res.end(JSON.stringify({status : 1}));
22. }
23. }).listen(1337,'127.0.0.1');
25. server.on('error',**function**(e){
26. **if**(e.code == 'EADDRINUSE'){
27. console.log('服务器地址及端口已被占用');
28. }
29. })
30. //设置服务器超时时间为1分钟
31. server.setTimeout(60\*1000,**function**(socket){
32. console.log('服务器超时');
33. console.log(socket);
34. })
35. server.on('close',**function**(){
36. console.log('服务器已关闭');
37. })
39. //连接数据库
40. **function** connect(params){
41. **var** connection = mysql.createConnection({
42. host : 'localhost',
43. port : 3306,
44. database : 'test',
45. user : 'root',
46. password : ''
47. });
48. connection.connect(**function**(err){
49. **if**(err){
50. console.log('与mysql数据库建立连接失败');
51. }**else**{
52. console.log('与mysql数据库建立连接成功');
53. connection.query('insert into user set ?',{
54. name : params.name,
55. sex : params.sex,
56. age : params.age,
57. tel : params.tel
58. },**function**(err,result){
59. **if**(err){
60. console.log('插入数据失败');
61. }**else**{
62. console.log('插入数据成功');
63. connection.end();
64. }
65. })
66. }
67. })
68. }

使用连接池连接数据库

**[javascript]** [view plain](http://blog.csdn.net/kongjunchao159/article/details/50232741" \o "view plain) [copy](http://blog.csdn.net/kongjunchao159/article/details/50232741" \o "copy)

1. //使用连接池连接数据库
2. **var** pool = mysql.createPool({
3. host : 'localhost',
4. port : 3306,
5. database : 'test',
6. user : 'root',
7. password : ''
8. });
9. **function** connect(params){
10. pool.getConnection(**function**(err,connection){
11. **if**(err){
12. console.log('与mysql数据库建立连接失败');
13. }**else**{
14. console.log('与mysql数据库建立连接成功');
15. connection.query('insert into user set ?',{
16. name : params.name,
17. sex : params.sex,
18. age : params.age,
19. tel : params.tel
20. },**function**(err,result){
21. **if**(err){
22. console.log('插入数据失败');
23. }**else**{
24. console.log('插入数据成功');
25. //当一个连接不需要使用时，将其归还到连接池中
26. connection.release();
27. //关闭连接池
28. //pool.end();
29. }
30. })
31. }
32. })
33. //处理数据库服务器连接中断时的操作
34. pool.on('error',**function**(err){
35. **if**(err.code === 'PROTOCOL\_CONNECTION\_LOST'){
36. console.log('与mysql数据库之间的连接丢失');
37. //3秒后重新尝试连接数据库
38. setTimeout(**function**(){
39. connect();
40. },3000);
41. }**else**{
42. **throw** err;
43. }
44. })
45. }

测试：

把项目放在服务器环境下，node执行server.js，然后提交表单即可

注意：由于跨域问题，使用CORS解决方案

JSONP由于传输数据量较大，需要使用POST方式提交，所以使用CORS技术（跨域资源共享）

CORS定义一种跨域访问的机制，允许一个域上的网络应用向另一个域提交跨域AJAX请求，只需由服务器发送一个响应标头即可

例子：（PHP环境）

header("Access-Control-Allow-Origin:\*");  //\*表示允许任何域向我们的服务端提交请求

header("Access-Control-Allow-Origin:http://www.test.com");

//只允许指定的域

（NodeJS环境）

res.setHeader("Access-Control-Allow-Origin","\*");

执行时，可能提示cannot find moudle 'mysql'，请在项目目录下安装mysql模块即可：npm install mysql