一、实验介绍

1.1 实验内容

本课程将通过使用 Node.js 与 socket.io 搭建服务程序，配合 Angular.js 能够动态声明内容，使用 bootstrap 框架的方式完成一个简单的聊天室应用。socket.io 封装了 WebSocket ，良好的兼容性，可以在不同的浏览器和移动设备上构建实时应用，非常方便和人性化。本课程主要学习 socket.io 和 Angular.js。

1.2 实验知识点

- socket.io 的使用
- bootstrap 的使用
- Angular.js 基础

1.3 实验环境

- Node.js 6.x

1.4 适合人群

本课程难度一般，属于初级课程，适合具有 javascript 基础和 Node.js 基础的用户学习 socket.io 和 Angular.js。

二、开发准备

2.1 从 npm init 开始

首先新建一个文件夹 `louChat` ，作为项目的根目录，同时也是我们的项目名称，然后执行命令：

```
$ cd louChat && npm init
```

我们会看到出现许多配置项需要你输入，同时这一过程也是引导你创建一个 `package.json` 文件，完成配置项填写后，此文件就存在根目录 `/louChat` 下了。

三、实验步骤

3.1 服务端主程序文件 app.js

项目需要 Express 框架托管服务程序，还需要 `socket.io` 建立实时服务，首先安装 Express 和 socket.io，在 `/louChat` 执行命令：

```
npm install --save express socket.io
```

在 `/louChat` 下新建 `app.js` 文件作为服务端主程序文件，敲入代码:

```
var express = require('express');
var app = require('express')();
var http = require('http').createServer(app);
var io = require('socket.io')(http);

// 设置静态文件路径
app.use(express.static(__dirname + '/client'));

app.get('/', function (req, res) {
    res.sendfile('index.html');
});

var connectedSockets={};
var allUsers=[{nickname:""}];
io.on('connection',function(socket){

    //有新用户进入聊天室
    socket.on('addUser',function(data){
        // coding there ...
    });

    //有用户发送新消息
    socket.on('addMessage',function(data){
        // coding there ...
    });

    //有用户退出聊天室
    socket.on('disconnect', function () {
        // coding there ...
    });
});

http.listen(3000, function () {
    console.log('app is running at port 3000 !');
});
```

在上面的代码中设置了三个事件监听代码：

- addUser
- addMessage
- disconnect

3.2 客户端

创建目录：

```
louChat
|--client
|  |--assets
|  |  |--js
|  |  |  `--client.js
|  |  `--css
|  |--index.html
|  |--message.html
|  `--user.html
`--app.js
```

`/louChat/client` 目录下是静态资源文件和客户端页面文件，其中的 `client.js` 就是客户端的负责监听服务端请求的文件。

3.2.1 使用 bootstrap

使用 bootstrap 快速实现布局良好的界面，首先执行命令 - `sudo npm install –g bower` 安装 `bower`，

在 `/louChat/client/assets` 目录下执行命令 - `bower install bootstrap` 即可

3.2.2 使用 Angular.js

在 `/louChat/client/assets` 目录下执行命令 - `bower install angular` 即可。

编辑 `/louChat/client/index.html` 文件：

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>mychat</title>
    <link href="/assets/bower_components/bootstrap/dist/css/bootstrap.min.css" rel="stylesheet">
    <link href="/assets/css/index.css" rel="stylesheet">
    <script src="/assets/bower_components/jquery/dist/jquery.min.js"></script>
    <script src="/socket.io/socket.io.js"></script>
    <script src="/assets/bower_components/bootstrap/dist/js/bootstrap.min.js"></script>
    <script src="/assets/bower_components/angular/angular.min.js"></script>
    <script src="./assets/js/client.js"></script>
</head>
<body ng-app="chatRoom" ng-controller="chatCtrl">

<div class="chat-room-wrapper" ng-show="hasLogined">
    <div class="online panel panel-success">
        <div class="panel-heading">
            <h3 class="panel-title">Online<span class="user-number">{{users.length-1}}</span></h3>
        </div>
        <div class="user-wrapper panel-body">
            <user iscurrentreceiver="receiver===user.nickname" info="user" ng-click="setReceiver(user.nicknam
e)" ng-repeat="user in users"></user>
        </div>
    </div>
    <div class="chat-room panel panel-success">
        <div class="panel-heading">
            <h3 class="panel-title">{{receiver?receiver:"Group-chat"}}</h3>
        </div>
        <div class="message-wrapper panel-body">
            <message self="nickname" scrolltothis="scrollToBottom()" info="message" ng-repeat="message in messa
ges"></message>
        </div>
        <div class="panel-footer">
            <form class="post-form form-inline" novalidate name="postform" ng-submit="postMessage()">
                <input type="text" class="form-control" ng-model="words" placeholder="say something~~"
required>
                <button type="submit" class="btn btn-success" ng-disabled="postform.$invalid">send</button>
            </form>
        </div>
    </div>
</div>

<div class="userform-wrapper"  ng-show="!hasLogined">
    <form class="form-inline login" novalidate name="userform" ng-submit="login()">
        <div class="form-group">
            <label for="nickname" class="sr-only"></label>
            <div class="input-group">
                <div class="input-group-addon"><span class="glyphicon glyphicon-user"></span></div>
                <input type="text" class="form-control" id="nickname" placeholder="your nickname" ng-model="nic
kname" required/>
            </div>
        </div>
        <button type="submit" class="btn btn-primary" ng-disabled="userform.$invalid">enter</button>
        <p ng-show="userExisted" class="help-block">This nickname already exists!</p>

    </form>
</div>

</body>
</html>
```

这里就需要将项目所需要的静态文件 jquery、 bootstrap、 Angular.js、 socket.io 等都引入的文件中，并且声明 Angular.js 的控制器范围。

user.html 文件：

```
<div class="user">
    <span class="nickname">{{info.nickname?info.nickname:"Group-chat"}}</span>
    <span class="unread" ng-show="info.hasNewMessage&&!iscurrentreceiver">[Unread]</span>
</div>
```

message.html 文件：

```
<div ng-switch on="info.type">
    <!-- 欢迎消息 -->
    <div class="system-notification" ng-switch-when="welcome"><strong>System:  </strong><span>{{inf
o.text}}</span>is coming~
    </div>
    <!-- 退出消息 -->
    <div class="system-notification" ng-switch-when="bye"><strong>System:  </strong>byebye,
        <span>{{info.text}}</span></div>
    <!-- 普通消息 -->
    <div class="normal-message" ng-class="{others:self!==info.from,self:self===info.from}" ng-switch-when="no
rmal">
        <div class="name-wrapper"><span>{{info.from}} @ </span><span>{{time |  date: 'HH:mm:ss' }}</span></di
v>
        <div class="content-wrapper"><span class="content">{{info.text}}</span></div>
    </div>
</div>
```

注：**CSS样式文件就由同学们自由发挥，这里就不展示出具体代码**

3.2.3 客户端主程序文件 client.js

在 `client.js` 文件中敲入以下代码：

```
var app=angular.module("chatRoom",[]);

app.factory('socket', function($rootScope) {
    var socket = io(); //默认连接部署网站的服务器
    return {
        on: function(eventName, callback) {
            socket.on(eventName, function() {
                var args = arguments;
                $rootScope.$apply(function() {
                    callback.apply(socket, args);
                });
            });
        },
        emit: function(eventName, data, callback) {
            socket.emit(eventName, data, function() {
                var args = arguments;
                $rootScope.$apply(function() {
                    if(callback) {
                        callback.apply(socket, args);
                    }
                });
            });
        }
    };
});

app.factory('userService', function($rootScope) {
    return {
        get: function(users,nickname) {
            if(users instanceof Array){
                for(var i=0;i<users.length;i++){
                    if(users[i].nickname===nickname){
                        return users[i];
                    }
                }
            }else{
                return null;
            }
        }
    };
});

app.controller("chatCtrl",['$scope','socket','userService',function($scope,socket,userService){
    var messageWrapper= $('.message-wrapper');
    $scope.hasLogined=false;
    $scope.receiver="";//默认是群聊
    $scope.publicMessages=[];//群聊消息
    $scope.privateMessages={};//私信消息
    $scope.messages=$scope.publicMessages;//默认显示群聊
    $scope.users=[];

    //登录进入聊天室
    $scope.login=function(){
        // coding there ...
    }
    $scope.scrollToBottom=function(){
       // coding there ...
    }

    $scope.postMessage=function(){
       // coding there ...
    }
    $scope.setReceiver=function(receiver){
        // coding there ...
    }

    // 收到登录结果
    socket.on('userAddingResult',function(data){
        // coding there ...
    });

    // 接收到欢迎新用户消息
    socket.on('userAdded', function(data) {
        // coding there ...
    });

    // 接收到在线用户消息
    socket.on('allUser', function(data) {
```

```
        // coding there ...
    });

    // 接收到用户退出消息
    socket.on('userRemoved', function(data) {
        // coding there ...
    });

    // 接收到新消息
    socket.on('messageAdded', function(data) {
        // coding there ...
    });
}]);
```

这里通过 Angular.js 的 `factory()` 定义了两个 service：

- socket
- userService

接着又声明了一个控制器，通过 `ng-controller` 指令可以操作 DOM。

3.3 登录

3.3.1 服务端

`/louChat/app.js` 文件：

```
//有新用户进入聊天室
socket.on('addUser',function(data) {
    if (connectedSockets[data.nickname]) {
        //昵称已被占用
        socket.emit('userAddingResult',{result:false});
    } else {
        socket.emit('userAddingResult',{result:true});
        socket.nickname=data.nickname;
        //保存每个socket实例,发私信需要用
        connectedSockets[socket.nickname]=socket;
        allUsers.push(data);
        //广播欢迎新用户,除新用户外都可看到
        socket.broadcast.emit('userAdded',data);
        //将所有在线用户发给新用户
        socket.emit('allUser',allUsers);
    }
});
```

注：不同 **`socket.emit()`** 的对比：

- `socket.emit()`： 信息传输对象为当前 socket 对应的 client，可看做一对一；
- `socket.broadcast.emit()`： 信息传输对象为所有的 client，排除当前 socket 对应的 client；
- `sockets.emit()`： 信息传输对象为所有的 client。

3.3.2 客户端

`../assets/js/client.js` 文件：

```
// 登录进入聊天室
$scope.login=function(){
    socket.emit("addUser",{nickname:$scope.nickname});
}
// 自动滚到最新信息的位置
$scope.scrollToBottom=function(){
    messageWrapper.scrollTop(messageWrapper[0].scrollHeight);
}

// here coding ...

//收到登录结果
socket.on('userAddingResult',function(data){
    if(data.result){
        $scope.userExisted=false;
        $scope.hasLogined=true;
    }else{//昵称被占用
        $scope.userExisted=true;
    }
});

//接收到欢迎新用户消息
socket.on('userAdded', function(data) {
    if(!$scope.hasLogined) return;
    $scope.publicMessages.push({text:data.nickname,type:"welcome"});
    $scope.users.push(data);
});

//接收到在线用户消息
socket.on('allUser', function(data) {
    if(!$scope.hasLogined) return;
    $scope.users=data;
});
```

3.4 显示在线人数

3.4.1 服务端

当用户登录成功后，服务端发送 addUser 的指令，指令中包含了所有用户信息的数组

3.4.2 客户端

../assets/js/client.js 文件：

```
app.directive('user', ['$timeout',function($timeout) {
    return {
        restrict: 'E',
        templateUrl: 'user.html',
        scope:{
            info:"=",
            iscurrentreceiver:"=",
            setreceiver:"&"
        }
    };
}]);
```

声明一个指令，将 user.html 页面嵌入 index.html

3.5 发送消息

3.5.1 服务端

/louChat/app.js 文件：

```
//有用户发送新消息
socket.on('addMessage',function(data){
    if(data.to){
        //发给特定用户
        connectedSockets[data.to].emit('messageAdded',data);
    }else{
        //群发-广播消息,除原发送者外都可看到
        socket.broadcast.emit('messageAdded',data);
    }
});
```

这里就可以运用之前说明的 `socket.emit` 和 `socket.broadcast.emit` 的不同来说实现不同的发送信息方式

3.5.2 客户端

`../assets/js/client.js` 文件:

```
$scope.postMessage=function(){
    var msg={text:$scope.words, type:"normal", from:$scope.nickname, to:$scope.receiver};
    var rec=$scope.receiver;
    if(rec){
       if(!$scope.privateMessages[rec]){
           $scope.privateMessages[rec]=[];
       }
        $scope.privateMessages[rec].push(msg);
    }else{
        $scope.publicMessages.push(msg);
    }
    $scope.words="";
    if(rec!==$scope.nickname) {
        socket.emit("addMessage", msg);
    }
}
$scope.setReceiver=function(receiver){
    $scope.receiver=receiver;
    if(receiver){
        if(!$scope.privateMessages[receiver]){
            $scope.privateMessages[receiver]=[];
        }
        $scope.messages=$scope.privateMessages[receiver];
    }else{
        $scope.messages=$scope.publicMessages;
    }
    var user=userService.get($scope.users,receiver);
    if(user){
        user.hasNewMessage=false;
    }
}

// here coding ...

// 接收到新消息
socket.on('messageAdded', function(data) {
    if(!$scope.hasLogined) return;
    if(data.to){ //私信
        if(!$scope.privateMessages[data.from]){
            $scope.privateMessages[data.from]=[];
        }
        $scope.privateMessages[data.from].push(data);
    }else{//群发
        $scope.publicMessages.push(data);
    }
    var fromUser=userService.get($scope.users,data.from);
    var toUser=userService.get($scope.users,data.to);
    if($scope.receiver!==data.to) {//与来信方不是正在聊天当中才提示新消息
        if (fromUser && toUser.nickname) {
            fromUser.hasNewMessage = true;//私信
        } else {
            toUser.hasNewMessage = true;//群发
        }
    }
});


// 新声明一个指令用于调用页面
app.directive('message', ['$timeout',function($timeout) {
    return {
        restrict: 'E',
        templateUrl: 'message.html',
        scope:{
            info:"=",
            self:"=",
            scrolltothis:"&"
        },
        link:function(scope, elem, attrs){
            scope.time=new Date();
            $timeout(scope.scrolltothis);
        }
    };
}]).directive('user', ['$timeout',function($timeout) {
        return {
        restrict: 'E',
        templateUrl: 'user.html',
        scope:{
            info:"=",
```

```
            iscurrentreceiver:"=",
            setreceiver:"&"
        }
    };
}]);
```

服务端对发送消息注入了两个方法 `postMessge` 和 `setReceiver`，对比查看 `index.html` 中对两个方法的使用，前者是发送消息，后者是指定发送消息的对象。

新增一个 **message** 指令，将 `message.html` 页面嵌入到 `index.html` 中。

3.6 退出

3.6.1 服务端

`/louChat/app.js` 文件：

```
//有用户退出聊天室
socket.on('disconnect', function () {
        //广播有用户退出
        socket.broadcast.emit('userRemoved', {
            nickname: socket.nickname
        });
        for(var i=0;i<allUsers.length;i++){
            if(allUsers[i].nickname==socket.nickname){
                allUsers.splice(i,1);
            }
        }
        //删除对应的socket实例
        delete connectedSockets[socket.nickname];
    }
);
```

3.6.2 客户端

`../assets/js/client.js` 文件：

```
//接收到用户退出消息
socket.on('userRemoved', function(data) {
    if(!$scope.hasLogined) return;
    $scope.publicMessages.push({text:data.nickname,type:"bye"});
    for(var i=0;i<$scope.users.length;i++){
        if($scope.users[i].nickname==data.nickname){
            $scope.users.splice(i,1);
            return;
        }
    }
});
```

这样整个课程的项目就完成了，运行项目：

```
$ node app.js
```

四、总结

本课程主要学习了 **socket.io** 和 **Angular**，项目中运用到了许多 **Angular.js** 的特性，这给项目带来了很好的体验，同时也给不熟悉 **Angular.js** 的同学带来了一定的学习压力。希望大家能通过参看官方资料，在问答区提问等方式学习。

前置课程

Angular.js实现即时搜索提示 (/courses/466)

Node.js 教程 (/courses/44)

进阶课程

Node.js 经典项目实战 (/courses/455)