web 应用实战——简易聊天室

吴梦堉　　计算机科学与技术学院
3180105091　　995862798@qq.com

日期：2020 年 6 月 7 日

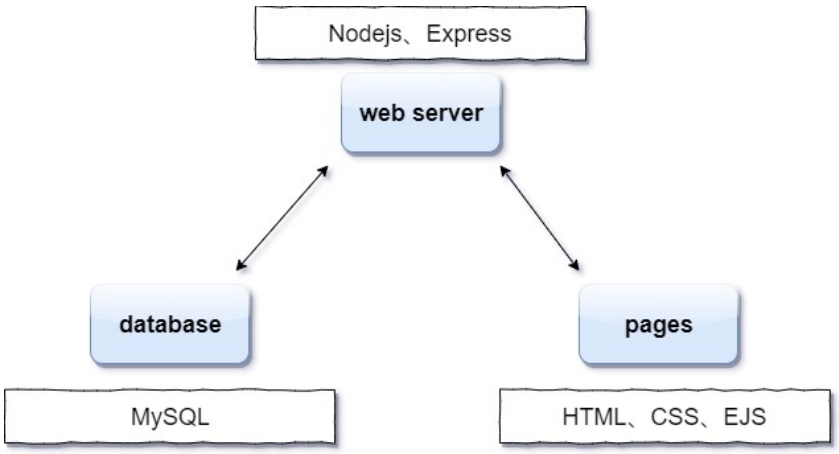
1　引言

1.1　背景

2020 年春夏学期秋冬课程 lab1.1&lab1.2 要求我们设计一款简易的 web 应用，以加强对于数据库、web 服务器、数据获取、页面设计等的学习与运用。

1.2　要求及技术栈

- 设计数据库结构及数据获取模型 -> MySQL
- 开发 web 服务器 -> Nodejs、express 框架
- 页面呈现设计与美化 -> HTML、CSS、EJS
- 运用 MVC 设计理念



2　应用介绍及使用方法

2.1　应用启动方法

- 一些必须的环境：
  Nodejs、npm 包管理工具、MySQL 数据库
- 安装依赖：
  npm install

- 导入数据库：

  将 sp_lab1.sql 导入到 MySQL 中
- 更改数据库连接的相关内容

  controller/controller.js 文件中第 8 行更改 MySQL 的 host、user、password 等信息
- 运行：

  npm run dev

  注意：请确保 3000 端口没被占用
- 使用：

  访问 localhost:3000 即可使用应用。

2.2 应用功能介绍

2.2.1 登陆注册界面

界面展示如图。点击加入我们、登入即可完成登陆和注册界面之间的转换

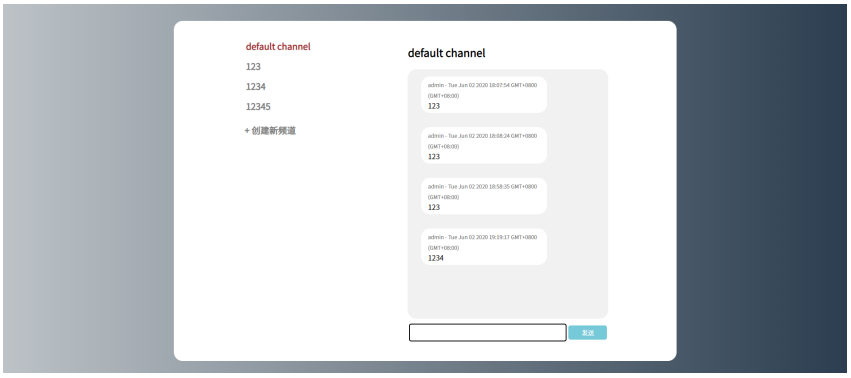
在登陆界面中，如果你勾选了维持登入状态，那么再未来的一段时间内，只要你不主动退出账号，你将保持已经登陆的状态。

我们为你提供了一个默认的账号：admin:admin

2.2.2 聊天室界面

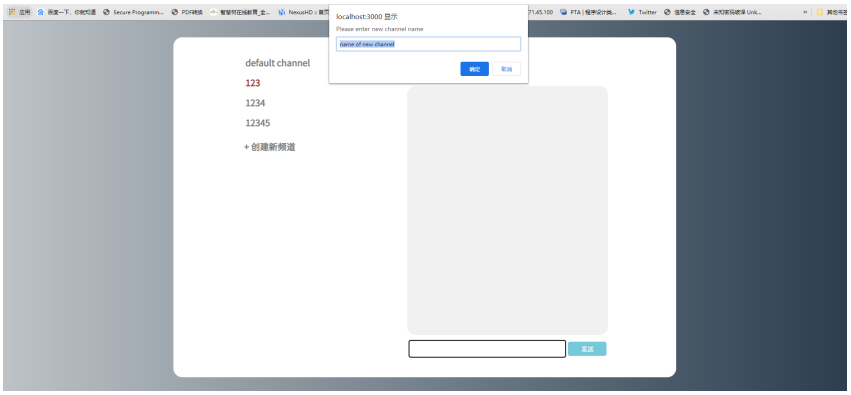
登陆或注册成功后，将进入到聊天室

聊天室界面如图

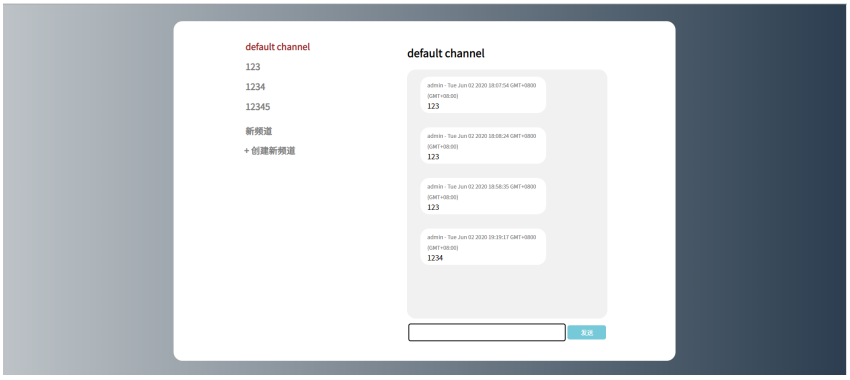


左侧为频道列表，显示当前已经存在的频道 (默认拥有一个名为 default channel 的频道)
右侧为聊天室，显示当前选中的频道内的聊天记录
点击频道名称可以更换频道，被选中的频道名称颜色将变为棕色。如下图，频道更换为 123。

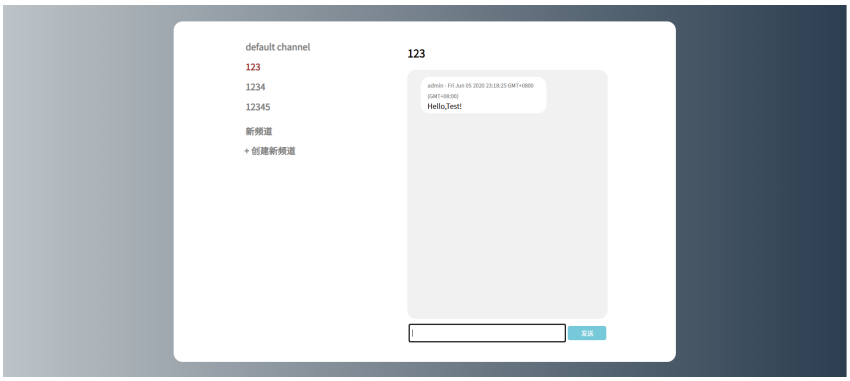
点击创建新频道，会弹出一个对话框，输入频道名称即可新建频道



频道创建成功



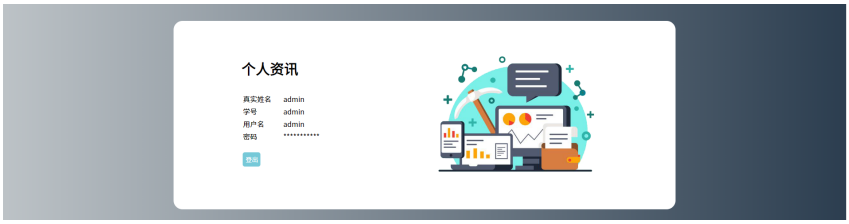
在聊天室中输入文字，点击发送即可实现信息发送。



聊天信息包含信息发送者名称、时间和信息内容。

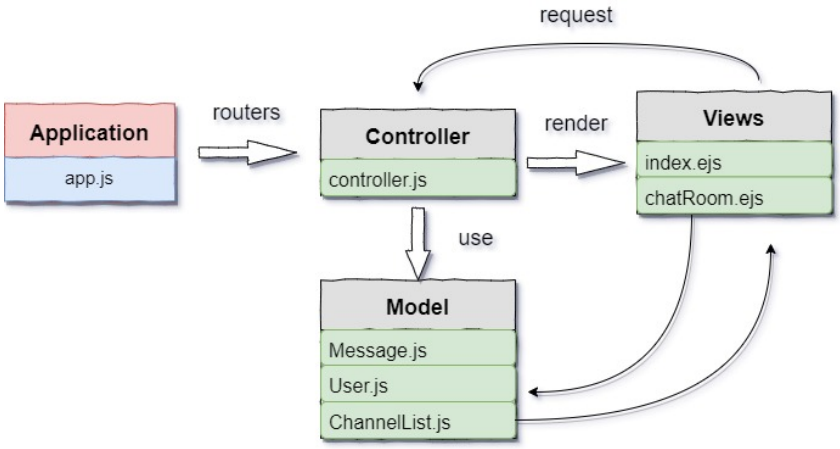
2.2.3 个人资讯

聊天室下方是个人资讯的展示。信息包括真实姓名、学号和用户名。由于密码是加密存储在数据库中的，因此无法展示。



3 应用设计详解

3.1 MVC 架构



3.2 Application

引入 express 模块作为后端的框架。使用 server 方法，在 3000 端口建立起服务器 server

```
const app = express();
const server = app.listen(3000, () => {
    console.log("\x1b[32m%s\x1b[0m", "[信息] 聊天室启动于 http://localhost
  :3000 ");
});
controller.server = server;
```

调用 session 和 cookie 模块，保存用户的登陆状态

```
app.use(cookieParser());
const MemoryStore = session.MemoryStore;
app.use(
    session({
        name: "app.sid",
        secret: "SecureProgrammingLab1@YDream",
        resave: true,
        saveUninitialized: true,
        cookie:
            ("name",
            "value",
            {
```

```
                maxAge: 5 * 60 * 1000,
                secure: false,
            }),
    })
);
```

设置 ejs 视图渲染引擎

```
app.set("view engine", "ejs");
```

router 分发。不同的路由调用 controller 对象的不同方法。这些方法的定义在 controller 中

```
//routers
app.get("/", (req, res) => {
    if (req.session && req.session.account) {
        res.render("chatRoom");
    } else {
        res.render("index");
    }
});
app.post("/login", controller.login);
app.post("/register", controller.register);
app.post("/getchannellist", controller.getChannelList);
app.post("/addchannel", controller.addChannel);
app.post("/addmsg", controller.addMsg);
app.post("/getchat", controller.getChat);
app.post("/changeChannel", controller.changeChannel);
app.post("/logout", controller.logout);
app.get("/chat", (req, res) => {
    res.render("chatRoom");
});
```

3.3 Controller

调用 model 层的四个 model

```
const User = require("../model/User");
```

```
const DataBase = require("../model/DataBase");
const ChannelList = require("../model/ChannelList");
const Message = require("../model/Message");

```

定义 controller 对象。并初始化数据库成员和连接成员。

```
const controller = {};
//Create object of the database
controller.dataBase = new DataBase("localhost", "root", "123", "sp_lab1",
  controller.server);
controller.conn = controller.dataBase.conn;

```

接下来是各个 router 对应的回调函数的定义。这些函数对于不同的请求进行权限、格式等的判断以后，调用 model 层提供的数据库的操作方法来获取并操作数据。随即针对不同的请求进行不同的数据响应和存储。数据库的具体实现体现在 model 层中，controller 中主要包含数据操作的逻辑，以及对于视图层的影响。

以登陆举例

```
controller.login = async (req, res) => {
  const { body } = req;
  const thisUser = new User(controller.conn);
  const usr = await thisUser.findById(body.user);
  if (!usr) {
      res.json({
          code: 2,
          msg: "没有该用户，请先注册",
      });
      res.end();
  } else {
      const flag = thisUser.checkPwd(body.user, body.pwd);
      if (!flag) {
          res.json({
              code: 3,
              msg: "密码错误",
          });
          res.end();
```

```
        } else {
            const { usr_name, pwd, Tname, student_num } = usr;
            if (body.keepLogin === "on") {
                req.session.account = "True";
            }
            req.session.channel_name = "default channel";
            req.session.usr_name = usr_name;
            req.session.pwd = pwd;
            req.session.Tname = Tname;
            req.session.student_num = student_num;
            res.render("../views/chatRoom", {
                usr_name,
                pwd,
                Tname,
                student_num,
            });
        }
    }
};

```

获取登陆表单中的 data 以后，首先调用 User 的 findById 方法，查询该用户名是否被注册过。接着调用 User 的 checkPwd 方法来判断用户名和密码是否匹配。然后进行 session 的修改，以及对 view 层进行响应。

3.4 Model

3.4.1 数据库设计

共设计了三张表

- users 存储用户信息

  字段：id、user_name 、pwd 、Tname 、student_num
- channel 存储频道信息

  字段：id、channel_name
- messages 存储聊天信息

  字段：id、usr_name、msg、time、channel_name . 其中 usr_name 和 channel_name 分别是其他两个表的外键

3.4.2 model 层设计

DataBase 设计了数据库的连接与初始化

```
class DataBase {
    constructor(host,user,password,database,server) {
        this.conn = mysql.createConnection({
            host: host,
            user: user,
            password: password,
            database: database
        })
        this.conn.connect((err) => {
            if (err) {
                console.log("\x1b[31m%s\x1b[0m",'[错误] 数据库连线失败，请检查配置');
                console.log("\x1b[31m%s\x1b[0m",'[错误] 错误内容: ' + err.message);
                console.log("\x1b[31m%s\x1b[0m",'[错误] 服务器初始化失败，即将关闭 ...');
                server.close();
            } else {
                console.log("\x1b[32m%s\x1b[0m",'[信息] 数据库连线成功。');
            }
        })
    }
}

```

以 user model 为例介绍 model 层的设计理念

User model 定义了与用户相关的数据操作方法。这些方法直接对数据库进行操作，由 controller 调用。为了避免“回调地狱”，我们使用 async/await 来进行异步函数的操作。因此我们对于 express 的 sql 操作进行了一层 promise 包装。以便于 controller 的直接调用

```
findById(usr_name) {
    const sql = "SELECT * FROM `users` WHERE `usr_name`='" + usr_name + "';";
    return new Promise((resolve, reject) => {
        this.conn.query(sql, (err, results) => {
            if (err) {
```

```
                reject(err);
            } else {
                resolve(results[0]);
            }
        })
    })
}

```

3.5 Views

View 采用了 ejs 作为载体，以方便进行数值传递。值得注意的是，ejs 采用后端渲染的方法。即后端处理好界面渲染成功之后，再发送到前端。并不是直接发送静态 html 界面。

View 层更多的是图形、排版的设计，不做赘述。

View 层同后端的交互位于 static/js 下，其中定义了请求发送的方法以及对于响应做出的动作

4 测试

按照上文提到的各种功能的使用方法以此进行了测试，结果均在预期之内。可以看到数据库的数据被成功添加。

id	usr_name	msg	time	channel_name
1	admin	123	2020-06-	default channel
2	admin	123	2020-06-	default channel
3	admin	123	2020-06-	default channel
4	admin	1234	2020-06-	default channel
5	admin	Hello,Tes	2020-06-	123

id	channel_name
1	default channel
2	123
3	1234
4	12345

id	usr_name	pwd	Tname	student_num
1	admin	21232f29	admin	admin
5	123	202cb962	123	123