深 圳 大 学 实 验 报 告

课程名称：基于 Web 的编程

实验项目名称：留言板制作

学院：计算机软件学院

专业：软件工程

指导教师：尹剑飞

报告人：洪继耀 学号：2014150120 班级：02

一、 实验要求

请制作一个留言板。要求如下：

1. 制作的留言板需要在本地服务器上成功运行；
2. 页面效果如下图所示；
3. **发布和显示只要一个 HTML+CSS+JavaScript 页面**，页面最上方是发布内容区域；
   a) **最多输入 140 个单词，界面自动显示还剩下多少字符可输入。**
4. 页面下方列出已经发布的帖子，还需显示提交时间；
5. 留言数据存在服务器的数据库或浏览器的 localStorage 中，
   a) 同时保证服务器与浏览器 localStorage 中的数据同步。可设计不同的存储策略，如在浏览器的 localStorage 中存放最近的 N=5 条留言，而服务器存放所有留言。
   b) 这里的同步设计有一定的挑战性，因为可以多个人同时留言，而每个人同时又可以看到其它人的留言（无需浏览器刷新），可以考虑使用 WebSocket 或在页面后台的 JavaScript 中用 jQuery.getJSON 定时从 Web 后端获取数据.
   c) 留言可以更改，更改后的留言，前端自动上传服务器后台。
6. 数据库可使用 MongoDB 或 MySQL 等数据库。
7. 尽最大努力使得页面美观、易用。
8. **抄袭作业得 0 分！、在实验报告中贴大量源代码扣 20~50 分。**
9. **提交：　有效设计说明、源代码打包上传（无需目标代码）**

留言板效果图

二、 设计说明

布局说明

总体布局说明

1. 页面是**响应式设计**，可以适应多终端。

2. 页面具有**动态的背景图**，并且背景图**平滑切换。**

3. 网站**有一个小 icon**，这里采用 vue 的 icon，不自行制作。

4. 网站核心部分是一个居中的面板。

总体效果图

留言发布区设计说明

效果图

代码

- 首先留言发布区用了 bootstrap 的面板组件。其代码如下:（为了更清晰，简化过，只留下与界面设计有关的部分）

```
<div class="panel-heading">留言发布区</div>
<div class="panel-body">
    <form class="form-group">
        <div class="input-group">
            <span class="input-group-addon">
                <span class="glyphicon glyphicon-user"></span>
            </span>
            <input type="text" placeholder="您的 ID" class="form-control">
        </div>
        <textarea  spellcheck="false"  type="text" placeholder="输入留言.." class="for
m-control comment"  rows="5"></textarea>
        <span class="text-num-limit">还可以输入{{textNumLimit}}个字</span>
        <button type="submit" class="btn btn-sm btn-primary send-btn">发布</button>
    </form>
</div>
```

设计说明

- 头部是由.panel-heading 修饰的面板头，内容区是.panel-body 修饰的面板内容区，面板头用 css 重新修饰过背景颜色和字的位置

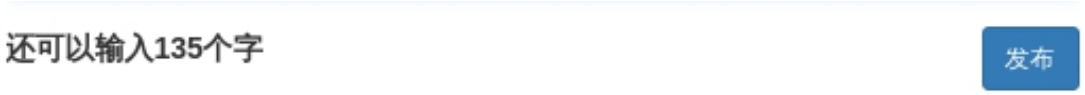
```
.panel-heading {
    text-align: center;
    border-top: 1px solid transparent;
    font-size: 1.2em;
    font-weight: bold;
    background-color: #eee !important;
}
```

- 面板内容区用了 bootstrap 的表单组件，所有的组件都被 form-control 修饰，具有宽度 100%的属性，从而实现响应式。
- 首先是.input-group 修饰的输入框组，由 input-group-addon 包裹的 icon 和一个 input 输入框组成。

- 接下来是一个 textarea，它用 spellcheck="false"取消了拼写检查，用 rows="5"

规定了有五行高，用 `word-break: break-all` 去掉了整词换行，用 `resize: none;` 去掉了右下角的改变大小的控件。

- 再接下来是用于显示计数的文字，和一个用 `btn btn-sm btn-primary` 分别修饰了形状大小颜色的按钮，他们在同一行。文字左浮动，按钮右浮动。



留言查看区设计说明

效果图

代码

- 依旧是一个面板组件，代码如下:（为了更清晰，简化过，只留下与界面设计有关的部分）

```
<div class="panel-heading">已经发布的留言</div>
<div class="panel-body">
    <div class="msg-box" transition="item">
```

```
        <img class="head-img">
        <div class="text-box">
            <h4 class="user-name"><strong>{{msg.userId}}</strong></h4>
            <button class="btn btn-sm btn-primary edit-btn" @click="editComment($inde
x)">编辑</button>
            <p class="user-comment">{{{msg.comment}}}</p>
            <form  class="form-group">
                <textarea  spellcheck="false"  type="text" placeholder="输入留言.." cl
ass="form-control comment" v-model="msg.comment" rows="5"></textarea>
                <span class="text-num-limit">还可以输入{{msg.textNumLimit}}个字</span>
                <button type="submit" class="btn btn-sm btn-primary send-btn">保存</bu
tton>
            </form>
            <p class="pull-left small text-info date">创建时间:{{msg.date}}</p>
            <p class="pull-left small text-info date">更新时间:{{msg.updateDate}}</p>
        </div>
    </div>
</div>
<div class="panel-heading">已无更多</div>

.msg-box {
  overflow: hidden;
  border-bottom: 1px solid #CCC;
  margin-bottom: 10px;
  position: relative;
  .edit-btn {
    position: absolute;
    top: 0px;
    right: 0px;
  }
  .head-img {
    float: left;
    width: 10%;
    margin: 10px 15px;
  }
  .text-box {
    float: left;
    width: 80%;
    word-break: break-all;
  }
  .date {
    width: 100%;
  }
}
```

设计说明

- 首先整个区域被两个 panel-heading 的块包住。

- 非编辑状态下，留言盒子部分左侧是一张左浮动的用.head-img 修饰的头像，占 10%。右侧是文字区域，由一个 h4 标签做 ID 显示，一个 p 标签做留言显示，两个一个 p 标签做时间显示，还有右上的用绝对定位定死的编辑按钮。每个盒子还有 1px 下边框做分割线。

- 编辑状态下，用作展示留言的 p 标签消失，变成一个 textarea，该区域几乎是上边发布区的翻版，故不加赘述。

- 编辑按钮与发布区的提交按钮一致，不加赘述。
- 时间部分有 `pull-left small text-info` 这三个 bootstrap 全局样式修饰，分别修饰位置，大小和颜色。

其他

说明

- 动画效果由 css3 和 js 共同控制。css3 主要实现过渡效果，而 js 主要修饰变化效果，还使用了 vue 提供的 `vue-animated-list` 插件，为留言的插入提供了动画效果。

代码

控制背景变换的过渡效果

```
body {
  transition: all 1s ease;
}
```

控制留言插入效果

```
<div v-for="msg in msgs" transition="item">
<script src="/javascripts/libs/vue-animated-list.js"></script>
.item-move {
  transition: transform .5s cubic-bezier(.55, 0, .1, 1);
}
```

概述

后端 API 是在 RESTful 风格的,发送和返回内容均为 json 数据。

API

留言相关

```
001 获取所有留言
GET /comment
Params
{
    null
}
Response
{
    code : '0',
    msgs : [{
        _id:            String, (mongodb 默认自动生成的主键)
        userId:         String, (用户 ID)
        comment:        String, (用户留言)
        headImgSrc:     String, (用户头像)
        date:           String, (留言创建时间)
        updateDate:     String, (留言更新时间)
        __v:            Number (mongodb 默认自动生成的版本锁标志，前端用不到)
    }]
}


Response Excetion : { code : '', msg : ''}
code    msg
-1     未知错误
```

002 上传留言

POST /comment

Params

```
{
    commentNew:
    {
        userId:     String, (用户 ID)
        comment:    String  (用户留言)
    }
}
```

Response

```
{
    code : '0'
}
```

Response Excetion ： { code : '', msg : ''}

code	msg
-1	未知错误

003 更新留言

PUT /comment

Params

```
{
    commentNew:
    {
        _id:        String, (从后端收到的留言 ID)
        comment:    String  (新的用户留言)
    }
}
```

Response

```
{
    code : '0',
    updateDate :    String  (更新时间)
}
```

Response Excetion ： { code : '', msg : ''}

code	msg
-1	未知错误

算法设计和分析

1.留言更新算法

代码

```
var pullMsgId = setInterval (()=> {
this.$http.get ('/comment').then (result => {
    if (result.data.code === '0') {
        var newMsgs = result.data.msgs;
        var newLength = newMsgs.length;
        var addLength = newLength - this.msgs.length;
        // 更新新的留言和触发插入动画效果
        for (var i = addLength - 1; i >= 0; i--) {
            newMsgs[ i ].textNumLimit = 140 - newMsgs[ i ].comment.length;
            newMsgs[ i ].editing = false; // 新增加两个只存在于前端的属性
            this.msgs.unshift (newMsgs[ i ]);
        }
        /* 更新修改过的留言 */
        for (var j = 0; j < this.msgs.length; j++) {
            if (this.msgs[ j ].editing === false) { // 不在编辑转态
            this.msgs[ j ].comment = newMsgs[ j ].comment;
            this.msgs[ j ].updateDate = newMsgs[ j ].updateDate;
            }
        }
        /* update LocalStorage*/
        var len = this.msgs.length > 20 ? 20 : this.msgs.length;
        var latestMsgs = this.msgs.slice (-len); // 深拷贝最旧的 20 条;
        latestMsgs=[];
        localStorage.setItem ('messages', JSON.stringify (latestMsgs));// 更新
    } else {
        console.log (result.data.code);
        }
    });
}, 1000);
```

算法说明

1. 留言更新算法可以用过 websocket 来实现，也可以用 ajax 轮询实现，这里采用定时 ajax 请求轮询后台，更新数据。

2. 首先用 setInterval()设置定时器通过 vue-resource 的$http.get()向后端发送

get 请求

3. 然后拿到数组之后，判断状态码是不是 0，是 0 进入下一步

4. 由于留言数组保存在 vue 的 vm 对象之中，所以这里可对比拿到的数组和现有的数组长度差异，然后根据长度差，把新的留言插入到现有数据中，用 unshift()方法,这个方法会让新的留言保持在数组的最前面，从而保持新的留言在前面，然后循环检测别人对旧留言的修改并覆盖

5. 更新 localStorage,用 localStorage.setItem()

6. 注意 localStorage 只能存字符串，因此要用 JSON.stringify()序列化

7. 为什么 4 中不将请求到的数组直接覆盖本地数组？因为我使用了 vue-animated-list 这个动画库，直接覆盖不触发动画效果，只有插入的时候触发动画效果

8. 关于 localStorage 存最旧的 20 条，是因为最旧的数据在数据库中相对位置不变，而如果是最新的 20 条，下次打开来就用不到 localStorage 里的数据了，因为不一定是最新的数据。当然要改成老师要求的存最新的数据也可以做到，只不过没有意义罢了。

2.多人写-多人读的数据同步算法

算法说明

1. 多人写同个留言时，mongoose 模块会根据__v 的值来为 document 加同步锁，因此多个人写的同一 document 会以最后一个请求为准

2. 多人写不同留言时，不互相冲突，写完会互相更新视图

3. 多人添加留言时，由于数据库多了一些 document，会在别人那边更新留言。

4. 之所以能做到 2 和 3，是因为 ajax 轮询更新视图的时候，会强制修正数据库来的数据和本地数据的差异，一切以数据库的为准

5. 更新完数据库会更新 localStorge

3.页面控件的事件处理逻辑

关于 vue

vue 是一个前端 mvvm 框架，有了它就可以省去 dom 操作，进行数据绑定和事件监听，从而方便地管理页面控件

举例说明

代码

```
<textarea v-model="comment"  v-on:input="inputComment" v-on:keydown="checkComment"></textarea>
```

- v-on

说明

v-on 用于绑定事件监听器。用法是 v-on:事件="表达式" 可简写为 @事件="表达式"

其中事件是 dom 事件。表达式可以是一个方法的名字或一个内联语句。触发这个元素上的 dom 事件时，相应表达式会被执行

比如这里触发 oninput 事件时，会从 this.vm 这个对象的 methods 属性里找到 inputComment 方法，然后执行它

v-on 有一些修饰符，如.stop 可阻止事件冒泡，.prevent 可以阻止默认行为(如表单提交会跳转)

- v-model

说明

v-model 用于表单的双向数据绑定，将表单数据绑定到 this.vm 这个对象的 data 对象里的属性。

这样就不需要用 dom 操作来获取 value 值或 innerHTML 值，大大方便了事件处理

4.前端代码的非面向对象组织的局限性分析

分析

关于这个，我想说 HTML 是标记语言，而 CSS 是样式表，这两个无关面向对象。

不过，HTML 可以通过 js 渲染的 html 模板语言如 pug、Mustache、Transparency 等，或者通过有组件化能力的前端 MVVM 框架，实现一定程度的组件化封装。

而 CSS 则可以通过预处理器 SASS 和 LESS 来实现一定程度上的组件化封装。（虽然说到底 html 和 css 的封装都是借助其他语言的。）

js 则是一种暧昧的面向对象语言（准确的说是**基于对象、面向原型链**），能够实现**封装**（通过闭包和组件化）和**继承**（通过原型链），比起传统的 OPP 语言，更加灵活。

本项目用的 vue.js 一款优秀的 js 框架，它通过数据绑定，将 html 页面上所有动态变化部分的数据绑定到一个 Vue 对象之中，从而实现简单的封装，不同的 Vue 对象之间互不影响。

借此，还可以与 ES6 结合，实现组件化、模板化，组件之间可以互相通信，从而在构建大型前端应用时提高代码的复用率。

Vue 对象简要分析

一个典型的 Vue 对象

```
var vm = new Vue ({
    el: '#app',
    data: {
        data : null
    },
    created() {
    },
    ready() {
    },
    destroyed () {
    }
    methods: {
        foo : function function_name(argument) {
            // body...
        }
    }
})
```

上面的代码是一个典型的 vm 对象，其中：

- el ： 用来绑定 html document
- data ： 与 html 绑定的数据将被放在这里。类似于其他 OOP 语言里的类属性。
- create()、ready()、destroyed() 这些被称为生命周期钩子,他们会在一个组件的不同生命周期被调用，类似与构造函数和析构函数。
- methods ： 这个对象存储了所有的方法，这些方法可以用于事件处理，也可以被生命周期钩子调用。

三、 实验结果截图

本地运行

```
hongjiyao_2014150120@ubuntu:~/programing base web/3/app$ npm start

> app@0.0.0 start /home/hongjiyao_2014150120/programing base web/3/app
> node app.js

listening at port 3000...
连接数据库成功!
```

主界面

添加留言

添加结果


已经发布的留言
洪继耀
编辑
发布了第二条留言
创建时间:2016年10月1日 22:49:39
更新时间:2016年10月1日 22:49:39
洪继耀
编辑
完成了作业。
很开心。
创建时间:2016年10月1日 22:48:33
更新时间:2016年10月1日 22:48:33
已无更多

编辑留言

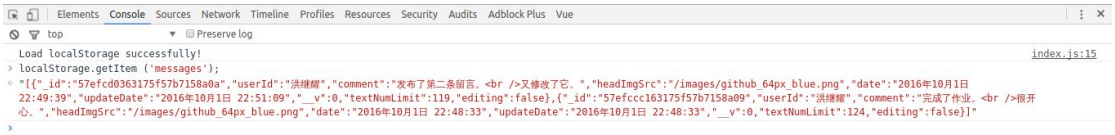

已经发布的留言
洪继耀
编辑
发布了第二条留言。
又修改了它。
还可以输入124个字
保存
创建时间:2016年10月1日 22:49:39
更新时间:2016年10月1日 22:49:39
洪继耀
编辑
完成了作业。
很开心。
创建时间:2016年10月1日 22:48:33

修改结果


已经发布的留言
洪继耀
编辑
发布了第二条留言。
又修改了它。
创建时间:2016年10月1日 22:49:39
更新时间:2016年10月1日 22:51:09
洪继耀
编辑
完成了作业。
很开心。
创建时间:2016年10月1日 22:48:33
更新时间:2016年10月1日 22:48:33
已无更多

此时的数据库

此时的 localStorage



四、 实验过程

（此处写你的过程，比如遇到的错误，以及解决方法）

遇到了...好像并没有遇到什么错误，由于上学期软件工程的大作业就是基于Bootstrap+vue+express，这个简化版的自然做得顺风顺水。

值得一提的一点，是关于 textarea 和 p 的一个坑，换行符在 textarea 里是’\n’，而在 p 标签内是”
”，如果要将 textarea 提交的东西 ajax 下载下来显示在 p 标签

里显示的话，要将’\n’正则匹配后转化成”
”才能正常显示换行。这里还有一个 vue 的不算坑的坑，就是 vue 的数据绑定语法{{}}会将 html 标签转化成 html 编码，从而防止 xss 攻击，也就是说如果使用{{}}将显示成 “行 1
行 2”，这时候应该换成{{{}}}来绑定数据，即可正常显示换行，虽然这样去掉了 xss 保护.....

做完之后第二天起床想到了更好的做法，即既使用 textarea 做表单提交，又使用 textarea 做信息显示，这时候只需要用 CSS 把用于显示的 textarea 边框和右下角的拉伸符号去掉，再把滚动条隐藏，即可模拟出 p 标签的外观。

五、 实验总结

（可能的进一步的改进方案、React 设计方案探讨、）

进一步的改进方案: 提供登录功能。

React 设计方案：由于本人使用的是和 react 同样是前端 mvvm 框架的 vue.js，故这里不做探讨。这两种框架都有数据绑定功能，其实差不多的。只不过 vue 更加轻量，学习曲线更加平滑。顺带一提，vue 2.0 于今天（10 月 1 号发布），真是振奋人心啊。

指导教师批阅意见：   成绩评定：   指导教师签字： 年　　月　　日
备注：

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。