HTML5新特性 -- Unit07

1.学子问答--用户登录实践

• xzqa_author数据表结构

字段名称	数据类型	是否为空	默认值	扩展	描述
id	MEDIUMINT UNSIGNED	NO		KEY AUTO_INCREMENT	主键
username	VARCHAR(30)	NO		UNIQUE	用户名
password	VARCHAR(32)	NO			密码,MD5加密
nickname	VARCHAR(30)	YES	NULL		用户昵称
avatar	VARCHAR(50)	NO	unamed.jpg		用户头像
article_number	MEDIUMINT	NO	0		发表的文章数量

• 用户登录的实现

第一步：修改 src/views/Login.vue，完成当用户单击按钮时实现登录的业务

示例代码如下：

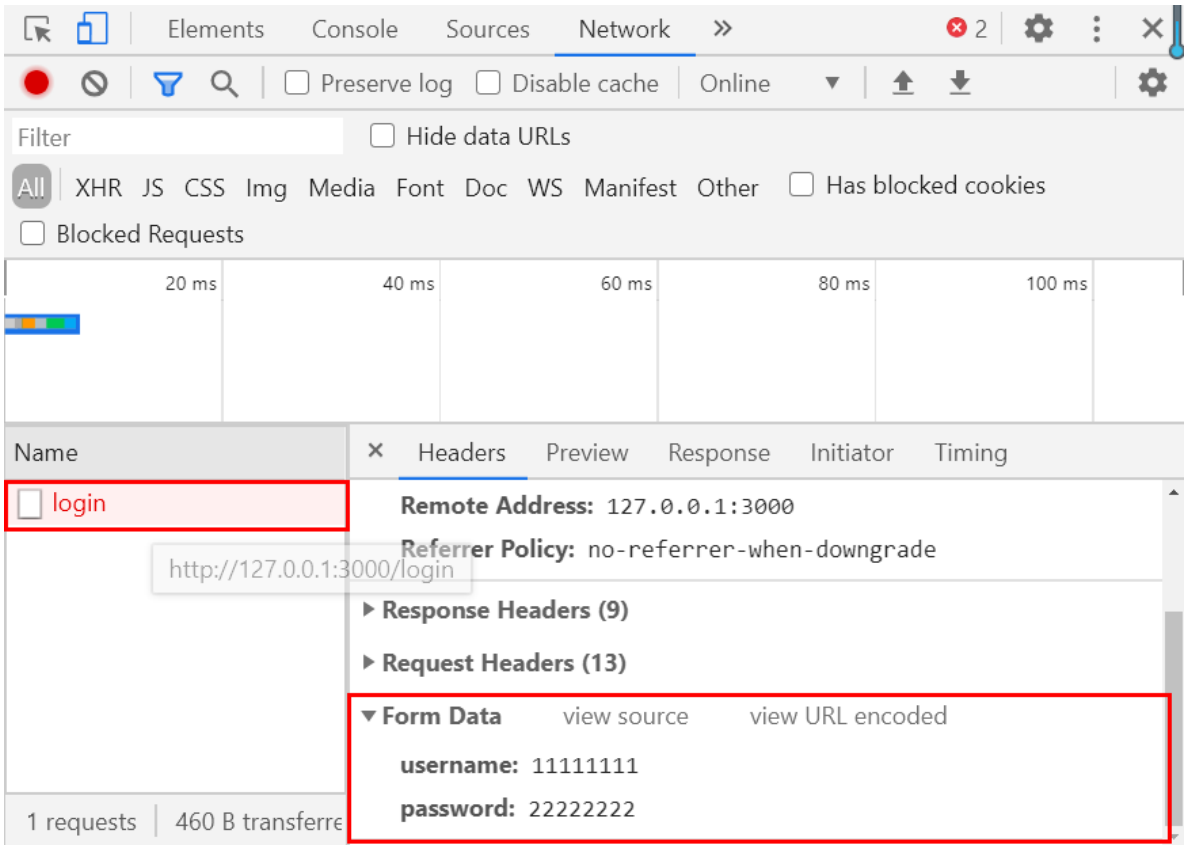
```
methods: {
    login() {
      if (this.checkUsername() && this.checkPassword()) {
        this.axios.post('/login').then(res=>{
            //
        });
      }
    }
}
```

以上代码存在两个问题：
第一：在发送 POST 请求时，没有将用户名和密码信息提交到 WEB 服务器

此时，上述代码修改如下：

```
methods: {
    login() {
      if (this.checkUsername() && this.checkPassword()) {
        this.axios.post('/login','username=' + this.username + '&password=' +
this.password).then(res=>{
            //
        });
      }
    }
}
```

此时运行的结果如下图所示：



第二：在`WEB`服务器上不存在对应的`API`路由

修改`WEB`服务器的`app.js`添加相关的路由，并且完成用户登录的业务逻辑，操作过程如下：

A.添加`POST`类型的`API`，名称为`/login`，示例代码如下：

```
// 用户登录的API
server.post('/login',(req,res)=>{
    //获取提交的用户名和密码并且以此为条件进行查找操作
});
```

B.因为`AJAX`请求为`POST`类型，那么必须通过`body-parser`中间件来获取`POST`提交的数据，那么也就意味着：

1. 安装`body-parser`中间件

```
npm install --save body-parser
```

2. 加载并且使用`body-parser`模块

```
//加载body-parser模块
const bodyParser = require('body-parser');

//创建Express应用
const server = express();

//使用body-parser中间件
server.use(bodyParser.urlencoded({
    extended:false
}));
```

C.以获取到的用户名和密码信息进行用户的查找操作，代码如下：

```
// 用户登录的API
server.post('/login',(req,res)=>{
    //获取提交的用户名和密码
    let username = req.body.username;
    let password = req.body.password;
    //并且以此为条件进行查找操作
    let sql = 'SELECT id,username FROM xzqa_author WHERE username=? AND
password=MD5(?)';
    pool.query(sql,[username,password],(err,results)=>{
        if(err) throw err;

        if(results.length == 0){
            res.send({message:'登录失败',code:0});
        }
        if(results.length == 1){
            //后续要调整
            res.send({message:'登录成功',code:1});
        }
    });
});
```

D.在`vue`客户端接收服务器响应信息，并且根据响应信息显示不同的提示内容，示例代码如下：

```
login() {
    if (this.checkUsername() && this.checkPassword()) {
        this.axios.post('/login','username=' + this.username +'&password=' +
this.password).then(res=>{
            if(res.data.code == 0){
                //Mint UI消息提示框
                this.$messagebox('登录失败');
            }
```

```
            if(res.data.code == 1){
                this.$router.push('/');
            }
        });
    }
}
```

• 登录状态的保持

当用户成功登录后，将影响到首页的导航的变化，其实也应该去影响其它页面的变化。也就需要将用户登录的状态保存到`Vuex`中，所以在`src/store/index.js`中修改如下：

```
state: {
    //标识用户是否登录
    isLogined:false
}
```

当存在以上代码之后，就可以在`src/views/Home.vue`中进行用户是否登录的判断了，如果已登录，则显示"注销"，否则显示"登录"及"免费注册"的信息。示例代码如下：

```
<!-- 顶部导航开始 -->
<mt-header title="学前端，到学问">
    <!-- 没有登录 -->
    <div slot="right" class="shortcut" v-if="!$store.state.isLogined">
        <router-link to="/register">免费注册</router-link>
        <router-link to="/login">登录</router-link>
    </div>
    <!-- 已登录 -->
    <div slot="right" v-else>
        <router-link to="/">
            <mt-button>
                <img src="../assets/images/logout.png" slot="icon">
            </mt-button>
        </router-link>
    </div>
</mt-header>
<!-- 顶部导航结束 -->
```

但是用户登录成功后，顶部导航没有发生任何改变，原因是：没有改变`state`的状态，所以

第一步：在`Vuex`中定义修改`isLogined`状态的`mutations`

```
mutations: {
    logined_mutations:(state)=>{
        //将用户登录状态改为真
        state.isLogined = true;
    }
}
```

第二步：在用户登录成功后，调用`mutations`以修改状态

在`src/views/Login.vue`调用`mutations`中的相关方法，示例代码如下：

```
login() {
    if (this.checkUsername() && this.checkPassword()) {
        this.axios.post('/login','username=' + this.username +'&password=' +
this.password).then(res=>{
            if(res.data.code == 0){
                //Mint UI消息提示框
                this.$messagebox('登录失败');
            }
            if(res.data.code == 1){
                //调用Vuex中的Mutations
                this.$store.commit('logined_mutations');
                this.$router.push('/');
            }
        });
    }
}
```

```
-- MySQL中更新记录
UPDATE 数据表名称 SET 字段名称 = 值[,字段名称 = 值,....] [WHERE 条件表达式]

-- 如,更新xzqa_author表中ID为15的记录,password字段值为 MD5(12345678)

UPDATE xzqa_author SET password=MD5('12345678') WHERE id=15;
```

现在可以通过正常的登录来实现首页顶部导航状态的变化了，但是 既使已登录的情况下，只要重新刷新页面，登录状态消失了 -- 只要刷新，页面将重新加载！所以应该将用户登录的状态进行保持 -- `WebStorage`，故：

第一步：当用户成功登录时，不仅要修改`state`中的状态还要将相关的信息写入到`webstorage`中，示例代码如下：

```
if(res.data.code == 1){
    //将用户登录的状态保存到webStorage中
    sessionStorage.setItem('isLogined',true);
    //修改Vuex中的state
    this.$store.commit('logined_mutations');
    this.$router.push('/');
}
```

第二步：当登录成功后，再次刷新页面，仍然出现状态丢失的情况 -- `Vuex` 中的 `state` 的初始值应该从 `webStorage` 中获取，如果获取不得到的话，则是 `false`

```
state: {
    //标识用户是否登录
    isLogined:sessionStorage.getItem('isLogined') ?
sessionStorage.getItem('isLogined')  : false
  }
```

第三步：经过上述代码，既使再刷新也可以保存用户登录状态了，但是无法实现"注销"操作，所示：

A、在 `Vuex` 中创建 `Mutations` 用于实现修改 `state` 中的状态，同时清理掉 `webStorage`

```
logout_mutations:(state)=>{
    //将用户登录状态改为真
    state.isLogined = false;
    //清理掉webStroage
    localStorage.clear();
}
```

B、单击"注销"按钮时，调用 `Mutations`

```
<div slot="right" v-else>
    <mt-button @click="logout">
        <img src="../assets/images/logout.png" slot="icon">
    </mt-button>
</div>
<script>
    export default{
        methods: {
            logout(){
                this.$store.commit('logout_mutations');
            }
        }
    }
</script>
```

2.webSocket

2.1 什么是webSocket、socket.io?

WebSocket是一个网络通信协议，其最大特点是：服务器可以主动向客户端推送消息，客户端也可以向服务器主动发送消息，是一种真正的平等的双向对话。

WebSocket协议于2008年诞生，2011年成为国际标准。

Socket.io是一个为浏览器与WEB服务器之间提供实时、双向和基于事件的软件通信库。其内部对websocket进行了封装，抹平了一些技术细节和平台的兼容性。

socket.io包括：

- Node.js服务器
- 浏览器--JavaScript客户端

2.2 WebSocket的优点

A.没有同源限制，客户端可以与任何服务器通信

B.数据格式比较轻量，通信高效

C.websocket协议的前缀是ws，如果加密的话，则为wss

2.3 安装 socket.io

2.3.1 服务器端

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

2.3.2 浏览器

A.下载Socket.io的JavaScript的客户端库文件 -- https://cdn.jsdelivr.net/npm/socket.io-client@2/dist/socket.io.js

B.直接在网页文件中通过<script>标签引用外部的JS文件