Lab 1

Lab1.1 Web Environment Setup-Java/Tomcat/Eclipse Overview

Web programming is a really big topic, which is related to a series of work. But In this lab, the only thing you have to do is: learn how to setup the web environment, include Java, Tomcat and Eclipse. It is the very base of all our further work about Web.

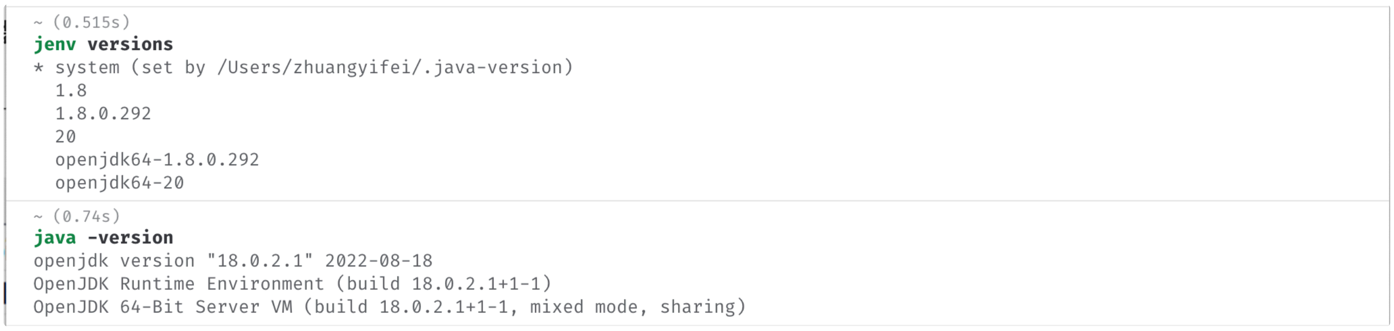
Before you start, you should choose the OS platform first, windows or linux. Actually, both platforms are suitable, just up to you!

Back to the lab, what’s our objective:

1. Setup the Java, and make sure it’s correct.
2. Setup the tomcat, and check the tomcat welcome page.
3. Setup the Eclipse, and learn how to use it.

Step 1 安装 JAVA

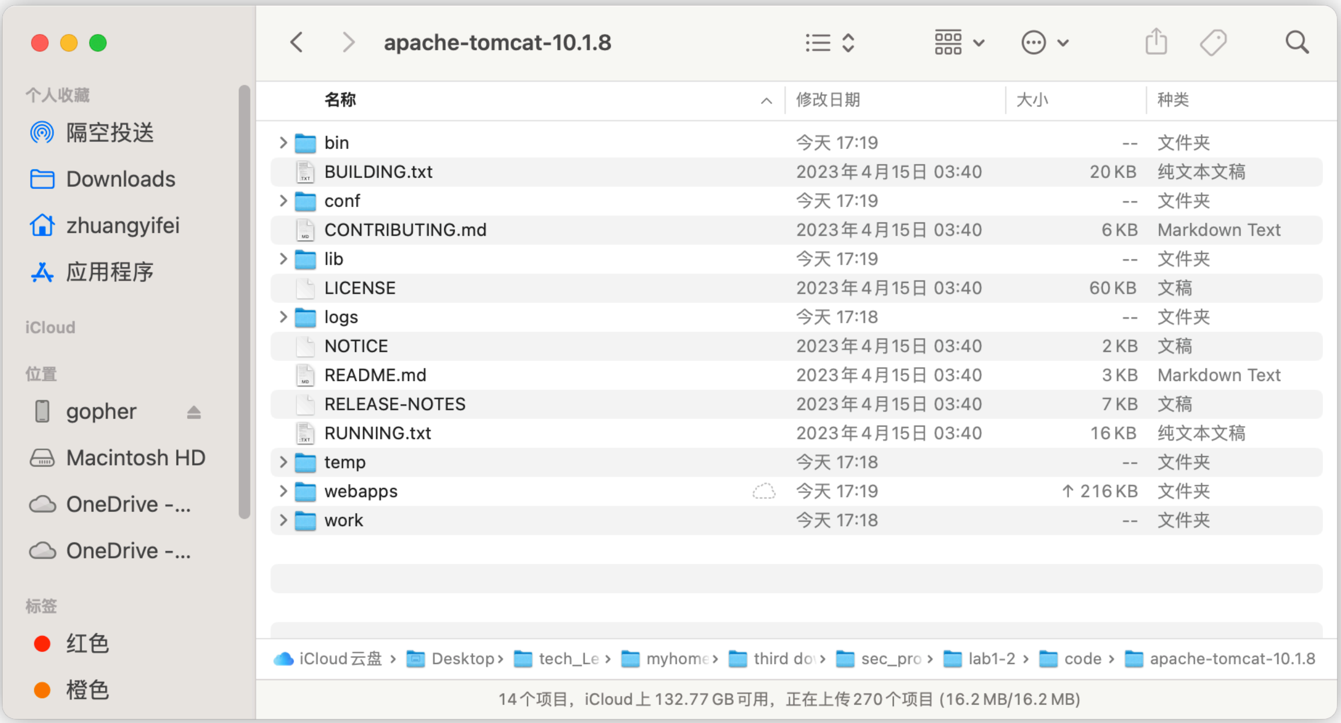
这里选择的操作系统是 macOS ，此前已经安装过 JAVA（使用 brew 安装），并且使用 jenv 进行管理。这里展示电脑中的 JDK 版本列表，以及实际使用的 JAVA
版本。



```
~ (0.515s)
jenv versions
* system (set by /Users/zhuangyifei/.java-version)
  1.8
  1.8.0.292
  20
  openjdk64-1.8.0.292
  openjdk64-20

~ (0.74s)
java -version
openjdk version "18.0.2.1" 2022-08-18
OpenJDK Runtime Environment (build 18.0.2.1+1-1)
OpenJDK 64-Bit Server VM (build 18.0.2.1+1-1, mixed mode, sharing)
```

Step 2 安装tomcat

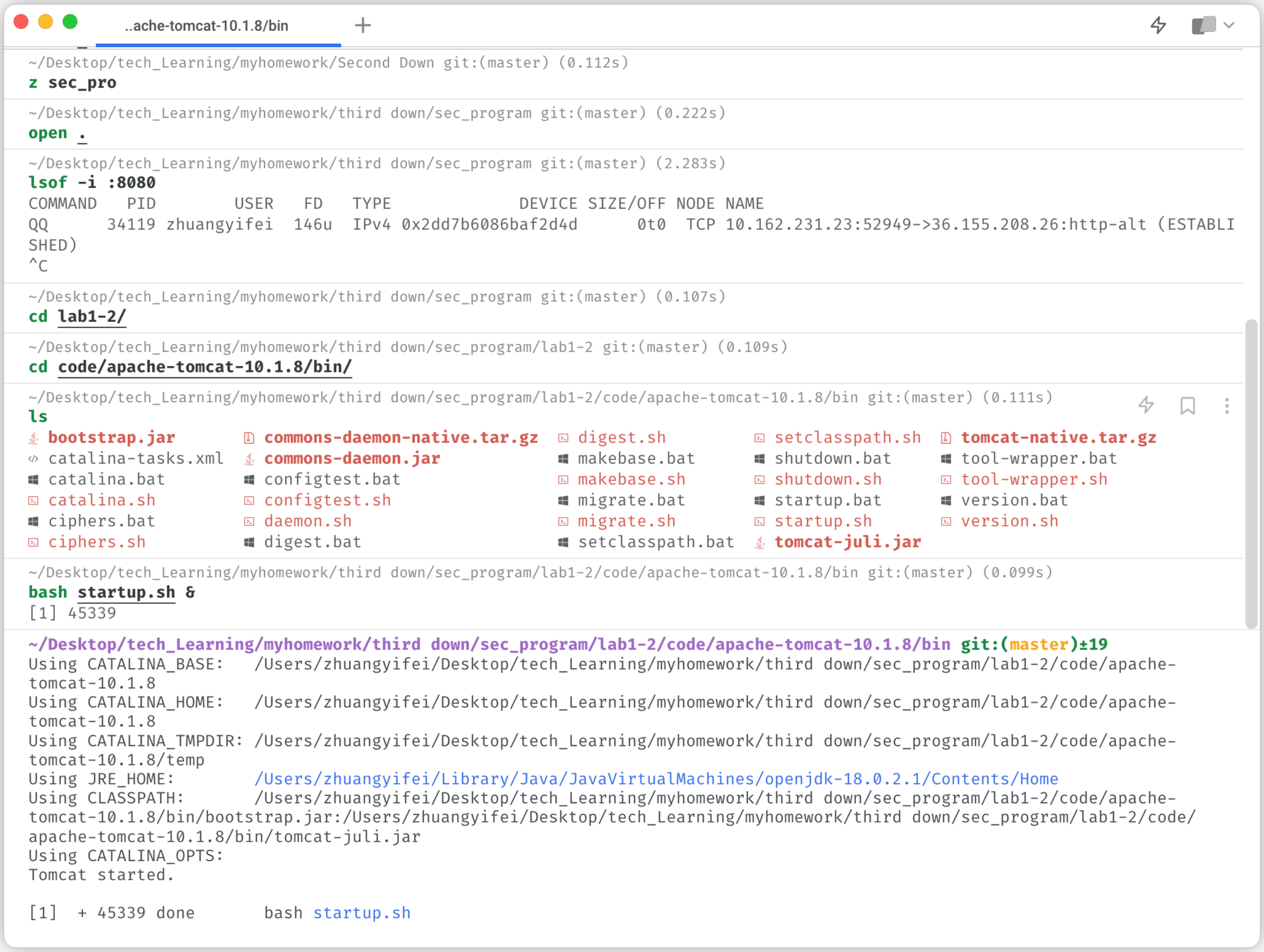
从 tomcat 官网下载压缩包，解压。



名称	修改日期	大小	种类
bin	今天 17:19	--	文件夹
BUILDING.txt	2023年4月15日 03:40	20 KB	纯文本文稿
conf	今天 17:19	--	文件夹
CONTRIBUTING.md	2023年4月15日 03:40	6 KB	Markdown Text
lib	今天 17:19	--	文件夹
LICENSE	2023年4月15日 03:40	60 KB	文稿
logs	今天 17:18	--	文件夹
NOTICE	2023年4月15日 03:40	2 KB	文稿
README.md	2023年4月15日 03:40	3 KB	Markdown Text
RELEASE-NOTES	2023年4月15日 03:40	7 KB	文稿
RUNNING.txt	2023年4月15日 03:40	16 KB	纯文本文稿
temp	今天 17:18	--	文件夹
webapps	今天 17:19	↑ 216 KB	文件夹
work	今天 17:18	--	文件夹

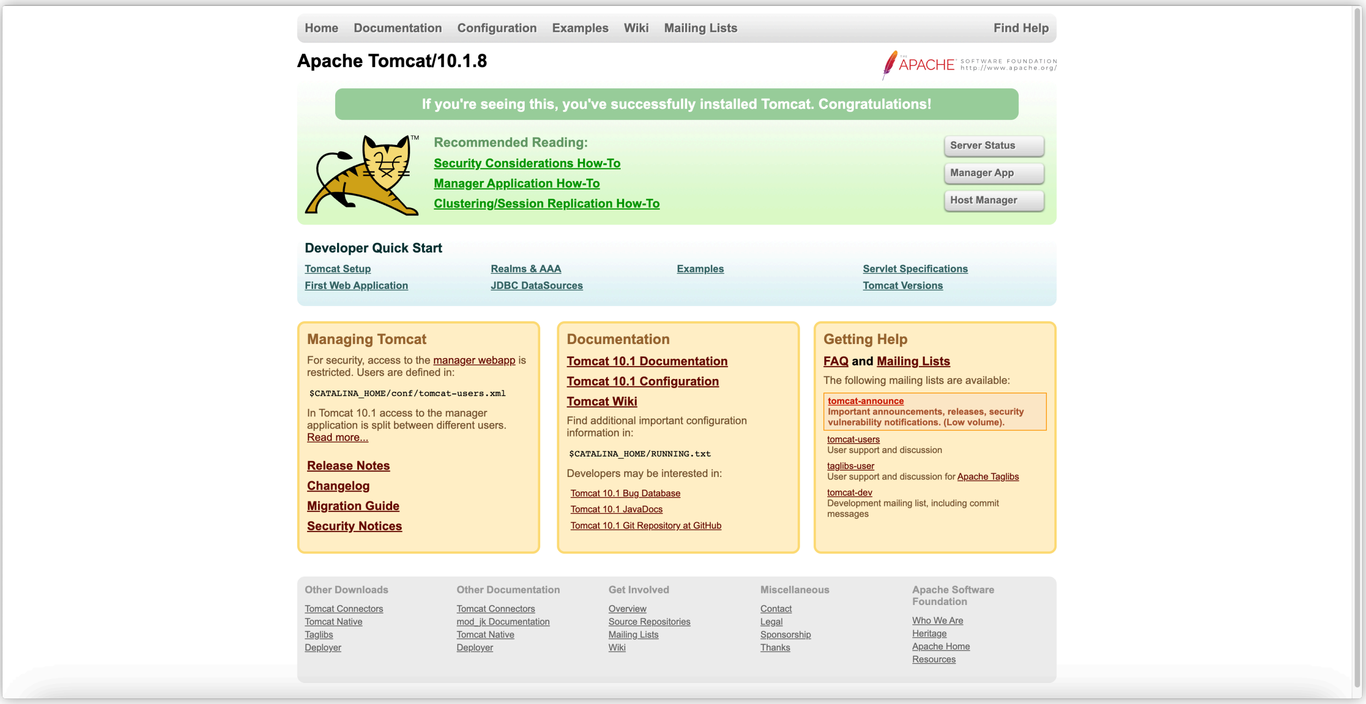
14个项目，iCloud上132.77 GB可用，正在上传270个项目 (16.2 MB/16.2 MB)

终端执行`bash startup.sh`命令，启动tomcat。



```
~/Desktop/tech_Learning/myhomework/Second Down git:(master) (0.112s)
z sec_pro

~/Desktop/tech_Learning/myhomework/third down/sec_program git:(master) (0.222s)
open .

~/Desktop/tech_Learning/myhomework/third down/sec_program git:(master) (2.283s)
lsof -i :8080
COMMAND   PID        USER   FD   TYPE             DEVICE SIZE/OFF NODE NAME
QQ      34119 zhuangyifei  146u  IPv4 0x2dd7b6086baf2d4d      0t0  TCP 10.162.231.23:52949->36.155.208.26:http-alt (ESTABLI
SHED)
^C

~/Desktop/tech_Learning/myhomework/third down/sec_program git:(master) (0.107s)
cd lab1-2/

~/Desktop/tech_Learning/myhomework/third down/sec_program/lab1-2 git:(master) (0.109s)
cd code/apache-tomcat-10.1.8/bin/

~/Desktop/tech_Learning/myhomework/third down/sec_program/lab1-2/code/apache-tomcat-10.1.8/bin git:(master) (0.111s)
ls
bootstrap.jar        commons-daemon-native.tar.gz  digest.sh          setclasspath.sh  tomcat-native.tar.gz
catalina-tasks.xml   commons-daemon.jar            makebase.bat       shutdown.bat     tool-wrapper.bat
catalina.bat         configtest.bat                makebase.sh        shutdown.sh      tool-wrapper.sh
catalina.sh          configtest.sh                 migrate.bat        startup.bat      version.bat
ciphers.bat          daemon.sh                     migrate.sh         startup.sh       version.sh
ciphers.sh           digest.bat                    setclasspath.bat   tomcat-juli.jar

~/Desktop/tech_Learning/myhomework/third down/sec_program/lab1-2/code/apache-tomcat-10.1.8/bin git:(master) (0.099s)
bash startup.sh &
[1] 45339

~/Desktop/tech_Learning/myhomework/third down/sec_program/lab1-2/code/apache-tomcat-10.1.8/bin git:(master)±19
Using CATALINA_BASE:   /Users/zhuangyifei/Desktop/tech_Learning/myhomework/third down/sec_program/lab1-2/code/apache-
tomcat-10.1.8
Using CATALINA_HOME:   /Users/zhuangyifei/Desktop/tech_Learning/myhomework/third down/sec_program/lab1-2/code/apache-
tomcat-10.1.8
Using CATALINA_TMPDIR: /Users/zhuangyifei/Desktop/tech_Learning/myhomework/third down/sec_program/lab1-2/code/apache-
tomcat-10.1.8/temp
Using JRE_HOME:        /Users/zhuangyifei/Library/Java/JavaVirtualMachines/openjdk-18.0.2.1/Contents/Home
Using CLASSPATH:       /Users/zhuangyifei/Desktop/tech_Learning/myhomework/third down/sec_program/lab1-2/code/apache-
tomcat-10.1.8/bin/bootstrap.jar:/Users/zhuangyifei/Desktop/tech_Learning/myhomework/third down/sec_program/lab1-2/code/
apache-tomcat-10.1.8/bin/tomcat-juli.jar
Using CATALINA_OPTS:
Tomcat started.

[1]  + 45339 done       bash startup.sh
```

浏览器访问`localhost:8080`，可以看到 tomcat 启动成功。



Home Documentation Configuration Examples Wiki Mailing Lists Find Help

Apache Tomcat/10.1.8

If you're seeing this, you've successfully installed Tomcat. Congratulations!

Recommended Reading:
Security Considerations How-To
Manager Application How-To
Clustering/Session Replication How-To

Server Status
Manager App
Host Manager

Developer Quick Start

Tomcat Setup
First Web Application
Realms & AAA
JDBC DataSources
Examples
Servlet Specifications
Tomcat Versions

Managing Tomcat

For security, access to the manager webapp is restricted. Users are defined in:

`$CATALINA_HOME/conf/tomcat-users.xml`

In Tomcat 10.1 access to the manager application is split between different users. Read more...

Release Notes
Changelog
Migration Guide
Security Notices

Documentation

Tomcat 10.1 Documentation
Tomcat 10.1 Configuration
Tomcat Wiki

Find additional important configuration information in:

`$CATALINA_HOME/RUNNING.txt`

Developers may be interested in:

Tomcat 10.1 Bug Database
Tomcat 10.1 JavaDocs
Tomcat 10.1 Git Repository at GitHub

Getting Help

FAQ and Mailing Lists

The following mailing lists are available:

tomcat-announce
Important announcements, releases, security vulnerability notifications. (Low volume).

tomcat-users
User support and discussion

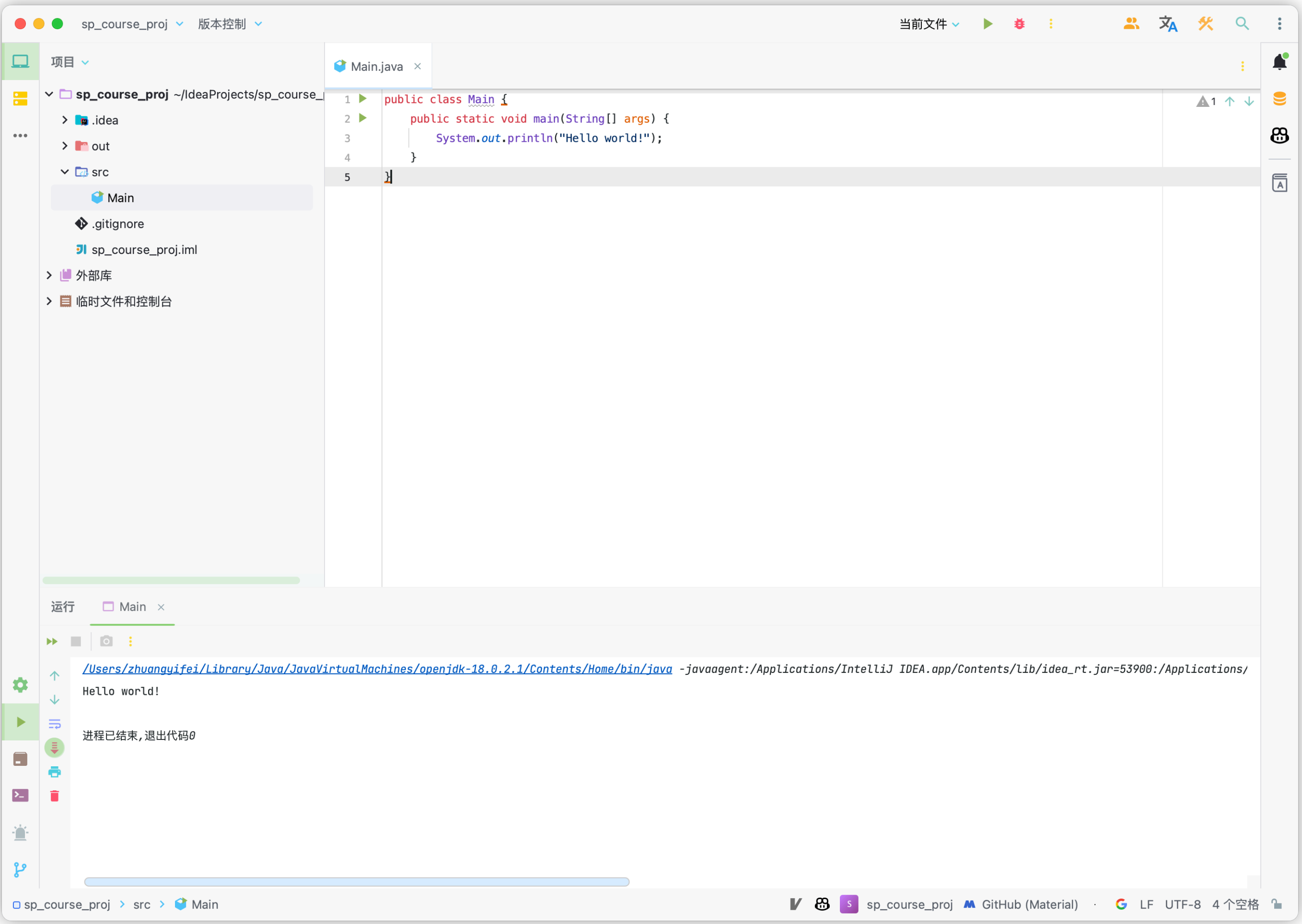
taglibs-user
User support and discussion for Apache Taglibs

tomcat-dev
Development mailing list, including commit messages

Other Downloads	Other Documentation	Get Involved	Miscellaneous	Apache Software Foundation
Tomcat Connectors	Tomcat Connectors	Overview	Contact	Who We Are
Tomcat Native	mod_jk Documentation	Source Repositories	Legal	Heritage
Taglibs	Tomcat Native	Mailing Lists	Sponsorship	Apache Home
Deployer	Deployer	Wiki	Thanks	Resources

Step 3 安装eclipse

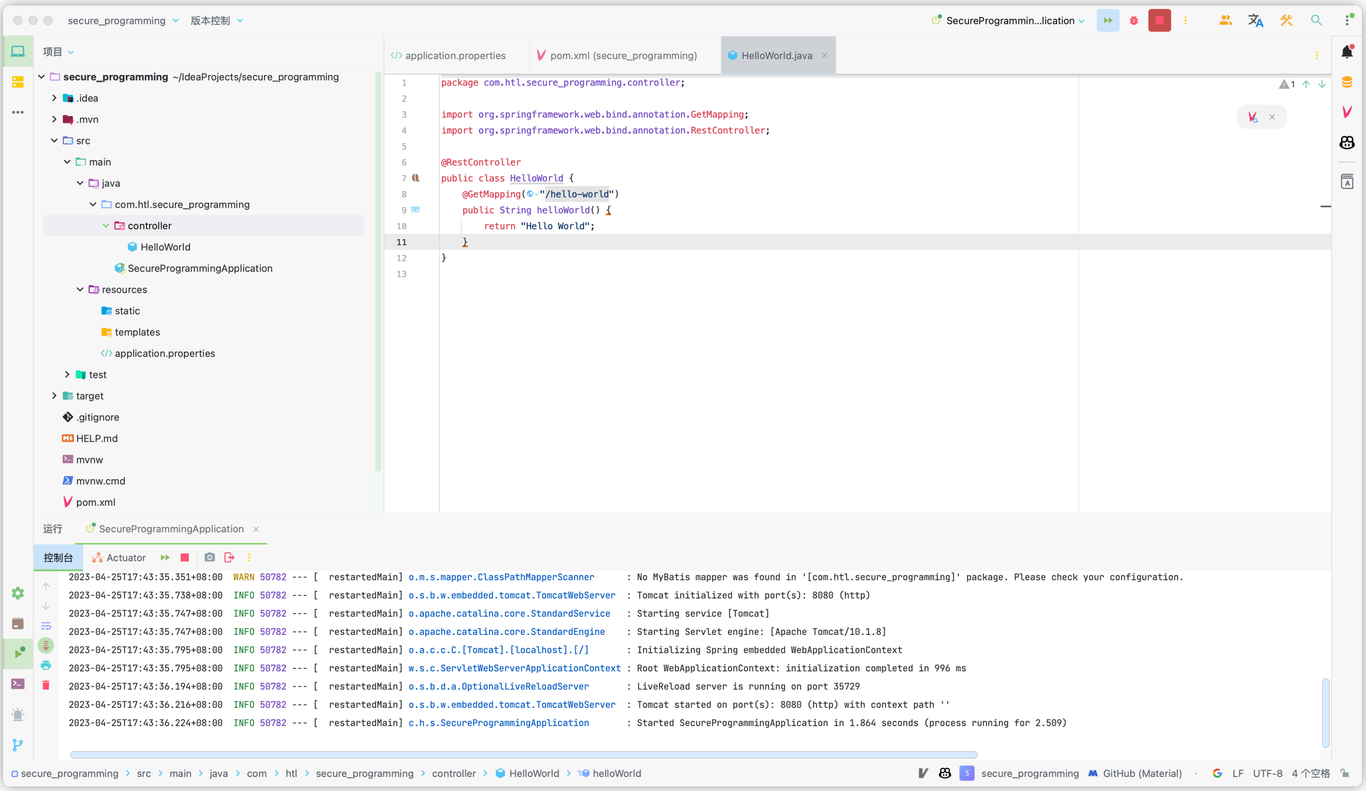
这里选择功能类似的Intellij IDEA进行操作，在 idea 中新建一个 JAVA 项目，然后创建一个HelloWorld项目，执行，以下是执行结果。



```java
public class Main {
    public static void main(String[] args) {
        System.out.println("Hello world!");
    }
}
```

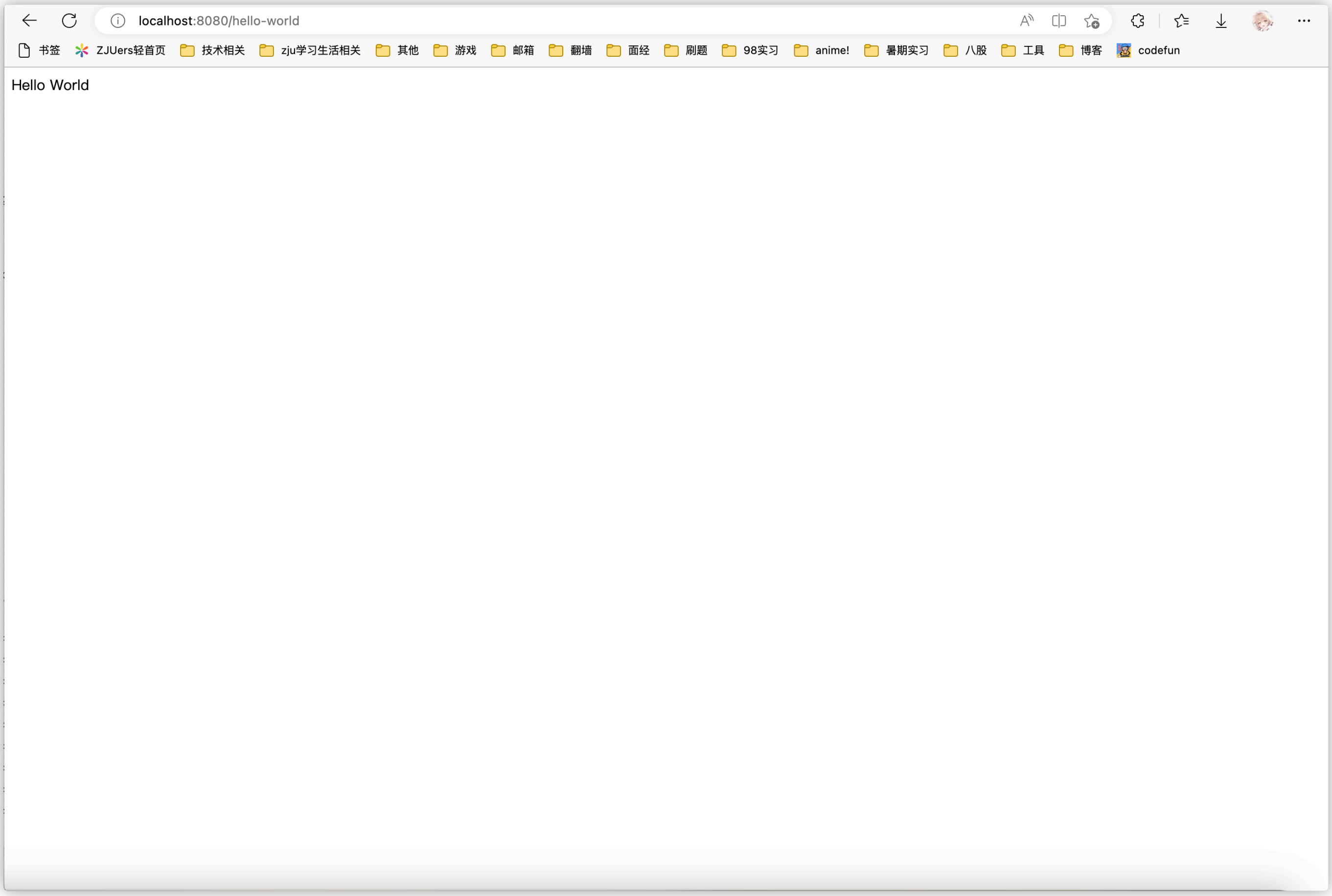
```
/Users/zhuangyifei/Library/Java/JavaVirtualMachines/openjdk-18.0.2.1/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA.app/Contents/lib/idea_rt.jar=53900:/Applications/
Hello world!

进程已结束,退出代码0
```

接下来，使用spring boot框架开发网页后端，首先创建一个Spring Boot项目，增加对应的Controller，然后编写返回Hello World的 Handler，之后启动项目，访问localhost:8080，即可看到Hello World的输出。



```java
package com.htl.secure_programming.controller;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class HelloWorld {
    @GetMapping("/hello-world")
    public String helloWorld() {
        return "Hello World";
    }
}
```

```
2023-04-25T17:43:35.351+08:00  WARN 50782 --- [  restartedMain] o.m.s.mapper.ClassPathMapperScanner      : No MyBatis mapper was found in '[com.htl.secure_programming]' package. Please check your configuration.
2023-04-25T17:43:35.738+08:00  INFO 50782 --- [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8080 (http)
2023-04-25T17:43:35.747+08:00  INFO 50782 --- [  restartedMain] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2023-04-25T17:43:35.747+08:00  INFO 50782 --- [  restartedMain] o.apache.catalina.core.StandardEngine    : Starting Servlet engine: [Apache Tomcat/10.1.8]
2023-04-25T17:43:35.795+08:00  INFO 50782 --- [  restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2023-04-25T17:43:35.795+08:00  INFO 50782 --- [  restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 996 ms
2023-04-25T17:43:36.194+08:00  INFO 50782 --- [  restartedMain] o.s.b.d.a.OptionalLiveReloadServer       : LiveReload server is running on port 35729
2023-04-25T17:43:36.216+08:00  INFO 50782 --- [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http) with context path ''
2023-04-25T17:43:36.224+08:00  INFO 50782 --- [  restartedMain] c.h.s.SecureProgrammingApplication       : Started SecureProgrammingApplication in 1.864 seconds (process running for 2.509)
```



localhost:8080/hello-world

```
Hello World
```

Lab 1.2 Implementation of Web Application

Overview

In lab 1.1, you have learned to setup the Web Programming Environment. It is easy, but very important, which is the base of lab 1.2.
In this lab, you will learn how to implement a web application. Usually, a web application is related with database, web server, data access, service, page design and so on. So there are really much you have to learn before you start the Implementation.
Back to the lab, what's our objective:

1. Design and access the database (MySQL recommended).
2. Learn to use web server and deploy your web application (Tomcat recommended).
3. Implement the application (Html, JSP recommended).
4. CSS is required to be used for controlling the page display in the “External Style Sheet” way.

Step 1 开发概述

我计划开发的项目是一个简单的 todolist，支持用户注册、登录、添加待办任务、标记任务状态和删除任务，项目使用前后端分离开发方式，后端使用spring boot开发，前端使用react + mui开发，开发过程如下。

1.1 技术选型

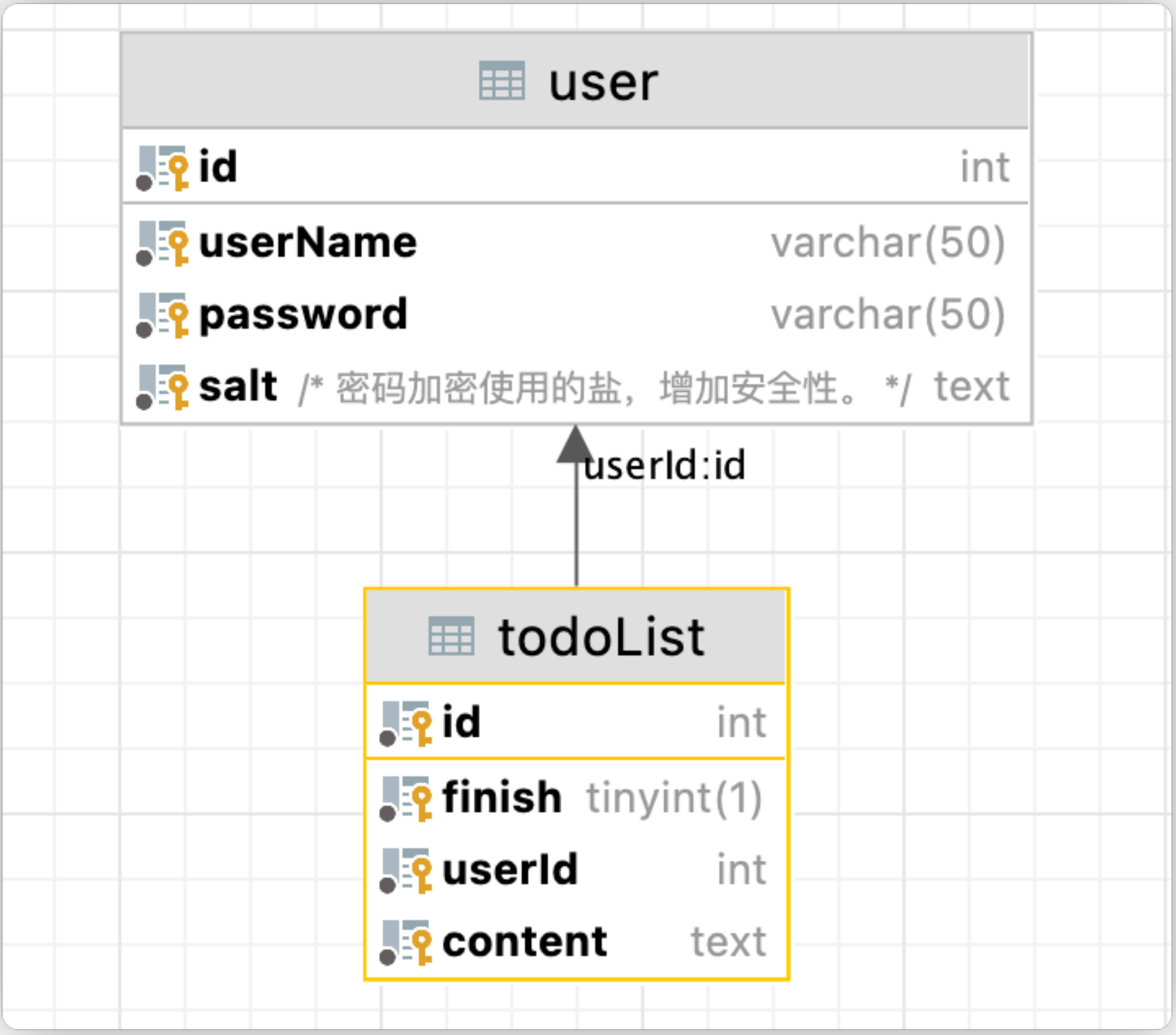
- 后端：
  - 数据库：MySQL
  - 开发语言：JAVA
  - 开发框架：Spring Boot + MyBatis + Spring MVC
- 前端
  - 开发语言：Javascript
  - 开发框架：React + Yarn + Material UI

Step 2 后端开发

2.1 设计数据表

项目主要有两个数据表，User表记录用户信息，todoList表负责记录用户的 todo 信息，表的 ddl 以及示意图如下。

```sql
create table user
(
    id       int auto_increment
        primary key,
    userName varchar(50) not null,
    password varchar(50) not null,
    salt     text        not null comment '密码加密使用的盐，增加安全性。',
    constraint user_userName_uindex
        unique (userName)
);

create table todoList
(
    id      int auto_increment
        primary key,
    finish  tinyint(1) default 0 not null,
    userId  int                  not null,
    content text                 null,
    constraint todoList_user_id_fk
        foreign key (userId) references user (id)
);


```

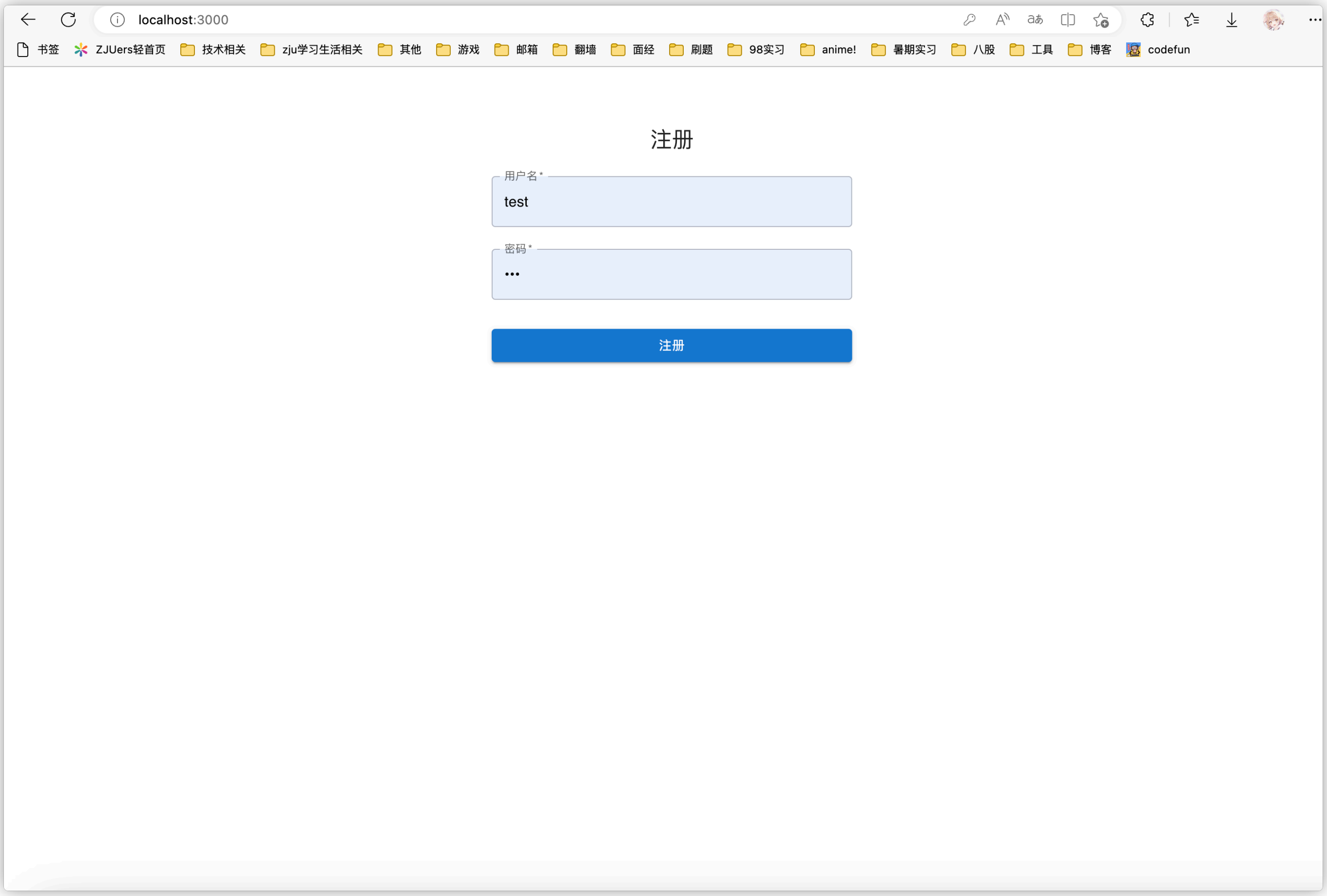


2.2 后端实现

2.2.1 注册实现

注册的时候，项目首先会进行参数校验，然后生成一个全局唯一的 salt，使用这个 salt 和 password 进行 md5 加密，将加密之后的结果和 salt 存储到数据库里面，避免数据库存储明文密码，保证安全性。

前端页面展示：



用户名 *
test

密码 *
•••

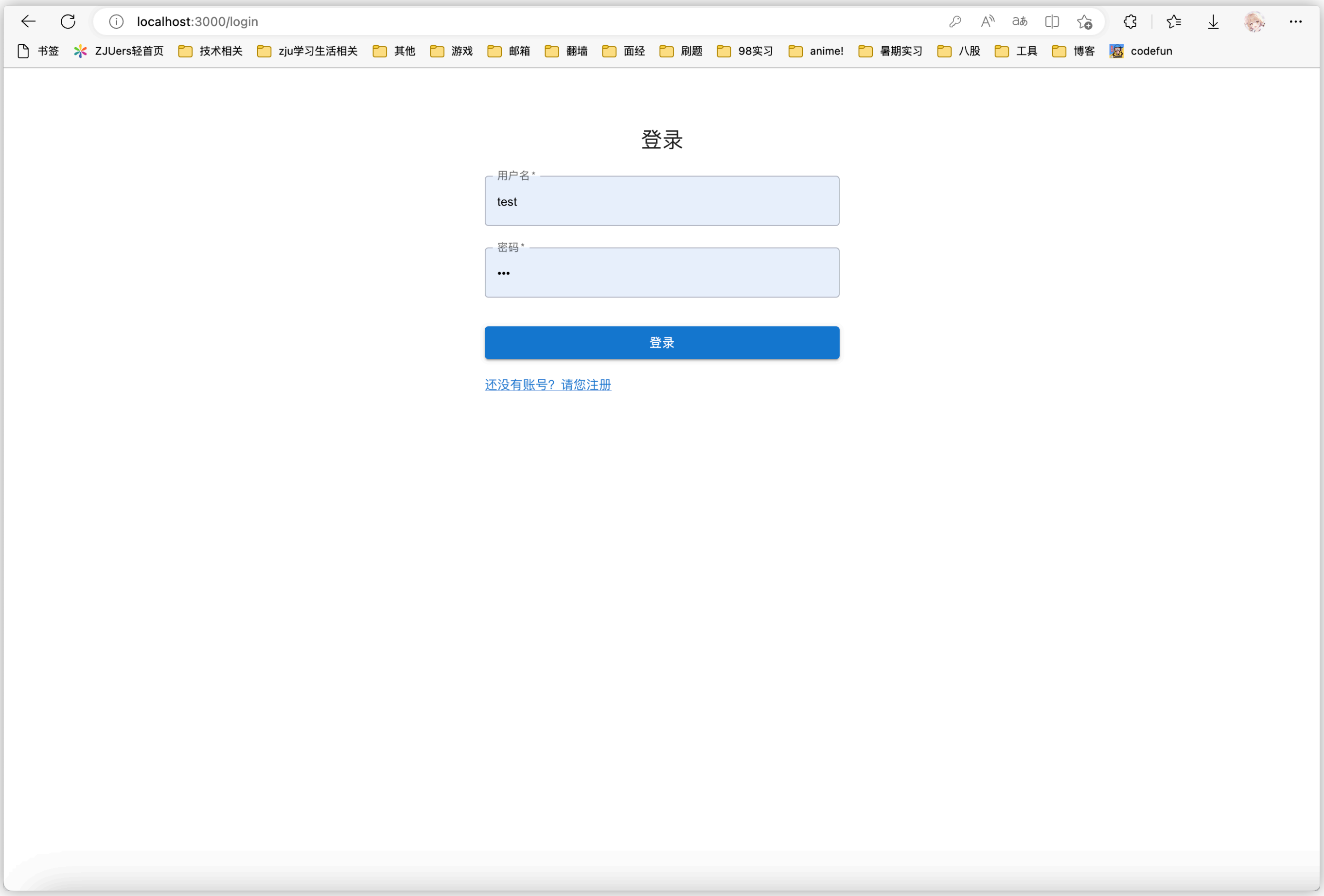
后端代码实现：

```java
/*UserController.java*/
@PostMapping("/register")
public String register(@Valid @RequestBody UserDTO userDTO, HttpServletResponse response) {
    UserPojo userPojo = new UserPojo(userDTO);
    try {
        userService.insert(userPojo);
        response.setStatus(200);
        return "success";
    } catch (Exception e) {
        e.printStackTrace();
        response.setStatus(HttpServletResponse.SC_INTERNAL_SERVER_ERROR);
        return e.toString();
    }
}
```

2.2.2 登录实现

登录的时候，系统会首先检查用户名和密码是否正确，如果错误，那么给前端返回unauthorize信号；如果正确，那么使用 JWT 技术生成一个令牌，然后将这个令牌返回给前端，表示登录成功。

前端页面展示：



登录代码：

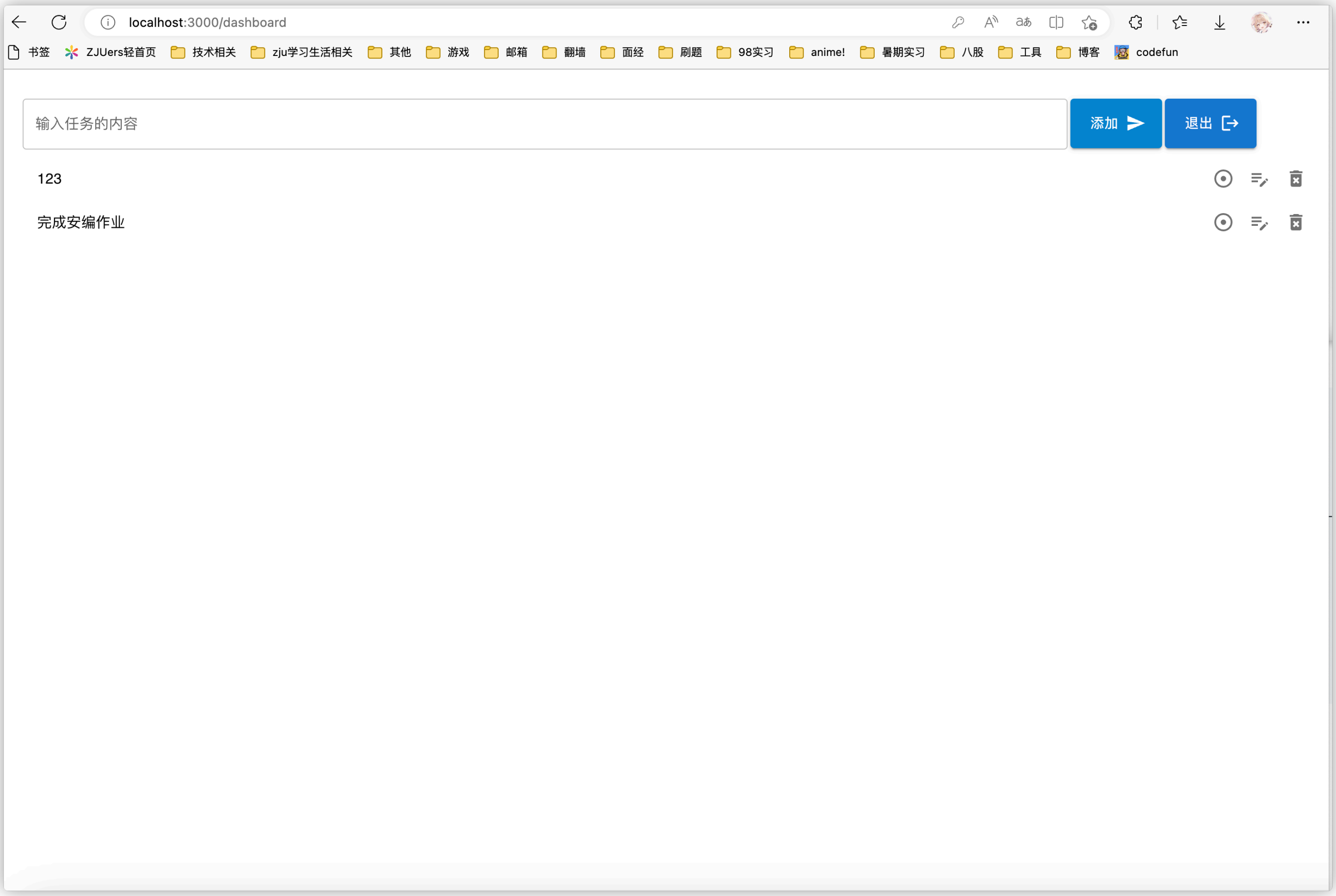
```java
@PostMapping("/login")
public String login(@Valid @RequestBody UserDTO userDTO, HttpServletResponse response) {
    try {
        if (userService.validate(userDTO.getUsername(), userDTO.getPassword())) {
            response.setStatus(200);
            return JwtUtil.createJWT(UUID.randomUUID().toString(), Math.toIntExact(userService.getUserByUserName(userDTO.getUsername()).getId()), (long) -1);
        }
        response.setStatus(HttpServletResponse.SC_UNAUTHORIZED);
        return "";
    } catch (Exception e) {
        e.printStackTrace();
        response.setStatus(HttpServletResponse.SC_INTERNAL_SERVER_ERROR);
        return "";
    }
}
```

JWT 模块相关代码

```java
package com.htl.secure_programming.utils;


import io.jsonwebtoken.Claims;
import io.jsonwebtoken.JwtBuilder;
import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.SignatureAlgorithm;
import org.springframework.stereotype.Component;

import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;
import java.io.IOException;
import java.util.Base64;
import java.util.Date;

@Component
public class JwtUtil {
    private static final String JWT_SECERT =
"asdhakjsgdjsavhjbbaskjchjakjsbcagghasdfavhfi2qy83rt1278figi13u2gf9812tv1ig32uohv12vy1ivg21io3hvo32y1i9v8t1g2vu9o32gh1v3g29fo13o9vg193fg1c98f89c1skd";


    public static String createJWT(String id, int userId, Long ttlMillis) throws
IOException {
        long nowMillis = System.currentTimeMillis();
        Date now = new Date(nowMillis);
        SecretKey secretKey = generalKey();
        JwtBuilder builder = Jwts.builder()
                .setId(id)
                .setSubject("" + userId)
                .setIssuer("htl")
                .setIssuedAt(now)
                .signWith(SignatureAlgorithm.HS256, secretKey);
        if (ttlMillis > 0) {
            long expMillis = nowMillis + ttlMillis;
            Date expDate = new Date(expMillis);
            builder.setExpiration(expDate);
        }
        return "Bearer " + builder.compact();
    }


    public static int validateJWT(String jwtStr) throws IOException {
        String userId = parseJWT(jwtStr).getSubject();
        return Integer.parseInt(userId);
    }

    private static SecretKey generalKey() throws IOException {
        Base64.Decoder decoder = Base64.getDecoder();
        byte[] encodedKey = decoder.decode(JWT_SECERT);
        return new SecretKeySpec(encodedKey, 0, encodedKey.length, "HmacSHA256");
    }


    public static Claims parseJWT(String jwt) throws IOException {
        SecretKey secretKey = generalKey();
        jwt = jwt.split(" ")[1].trim();
        try {
            return Jwts.parser()
                    .setSigningKey(secretKey)
                    .parseClaimsJws(jwt)
                    .getBody();
        } catch (Exception e) {
            e.printStackTrace();
            throw e;
        }
    }
}
```

2.2.3 待办事项添加

在登录成功之后，前端会跳转到待办事项界面，之后前端可以进行待办事项添加。
前端页面展示：


localhost:3000/dashboard

书签 ZJUers轻首页 技术相关 zju学习生活相关 其他 游戏 邮箱 翻墙 面经 刷题 98实习 anime! 暑期实习 八股 工具 博客 codefun

输入任务的内容

添加 退出

123

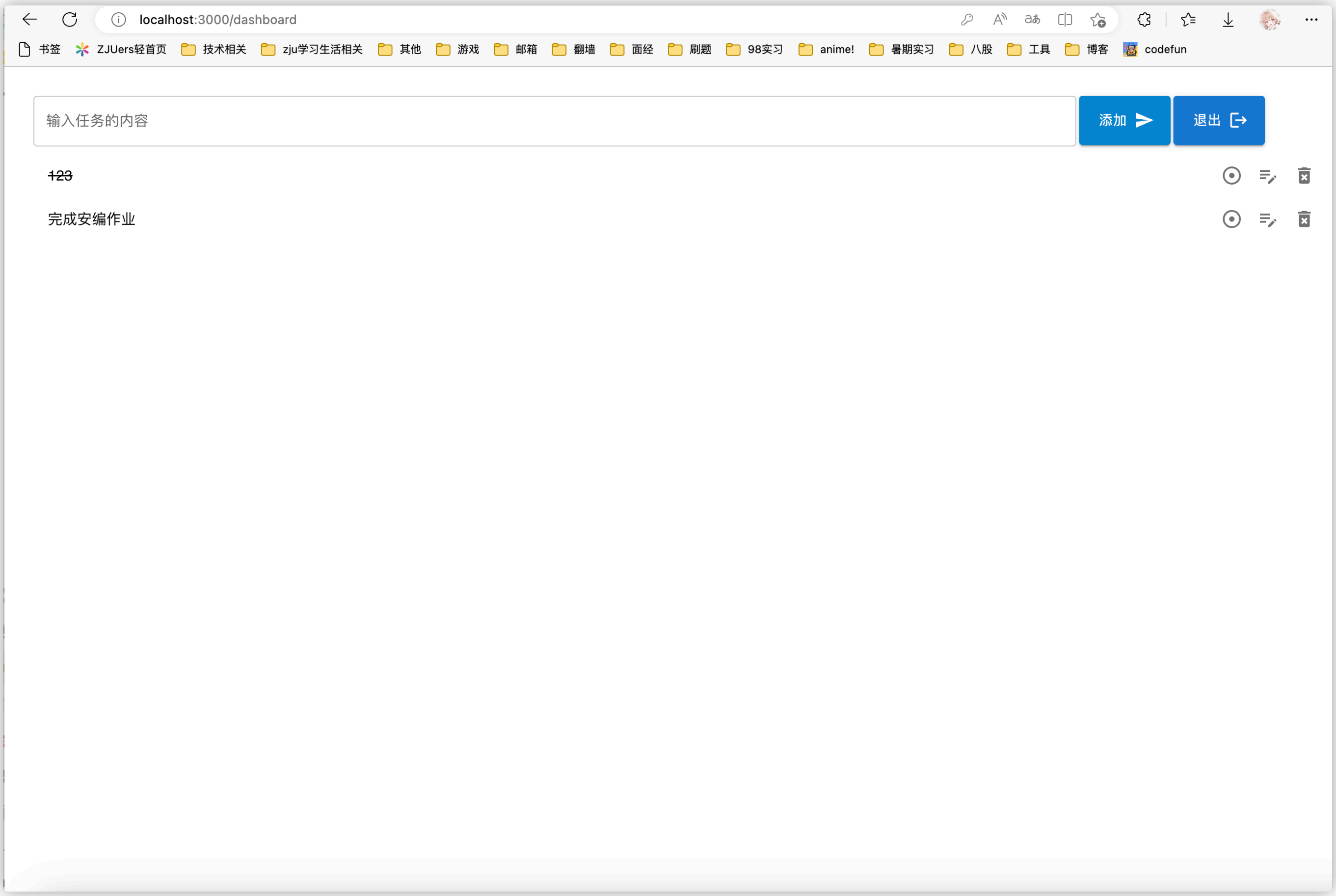
完成安编作业

后端代码：

```java
@PostMapping("/insert")
public String add(@RequestBody TodoListPojo todoListPojo, HttpServletResponse response) {
    try {
        int userId = getUserId();
        todoListPojo.setUserId(userId);
        response.setStatus(HttpServletResponse.SC_OK);
        todoListService.insert(todoListPojo);
        return "success";
    } catch (Exception e) {
        e.printStackTrace();
        response.setStatus(HttpServletResponse.SC_BAD_REQUEST);
        return e.toString();
    }
}
```

2.2.4 待办事项状态更新

点击事项上的第一个按钮，可以更新事项的完成状态，标记事项完成或未完成。



~~123~~

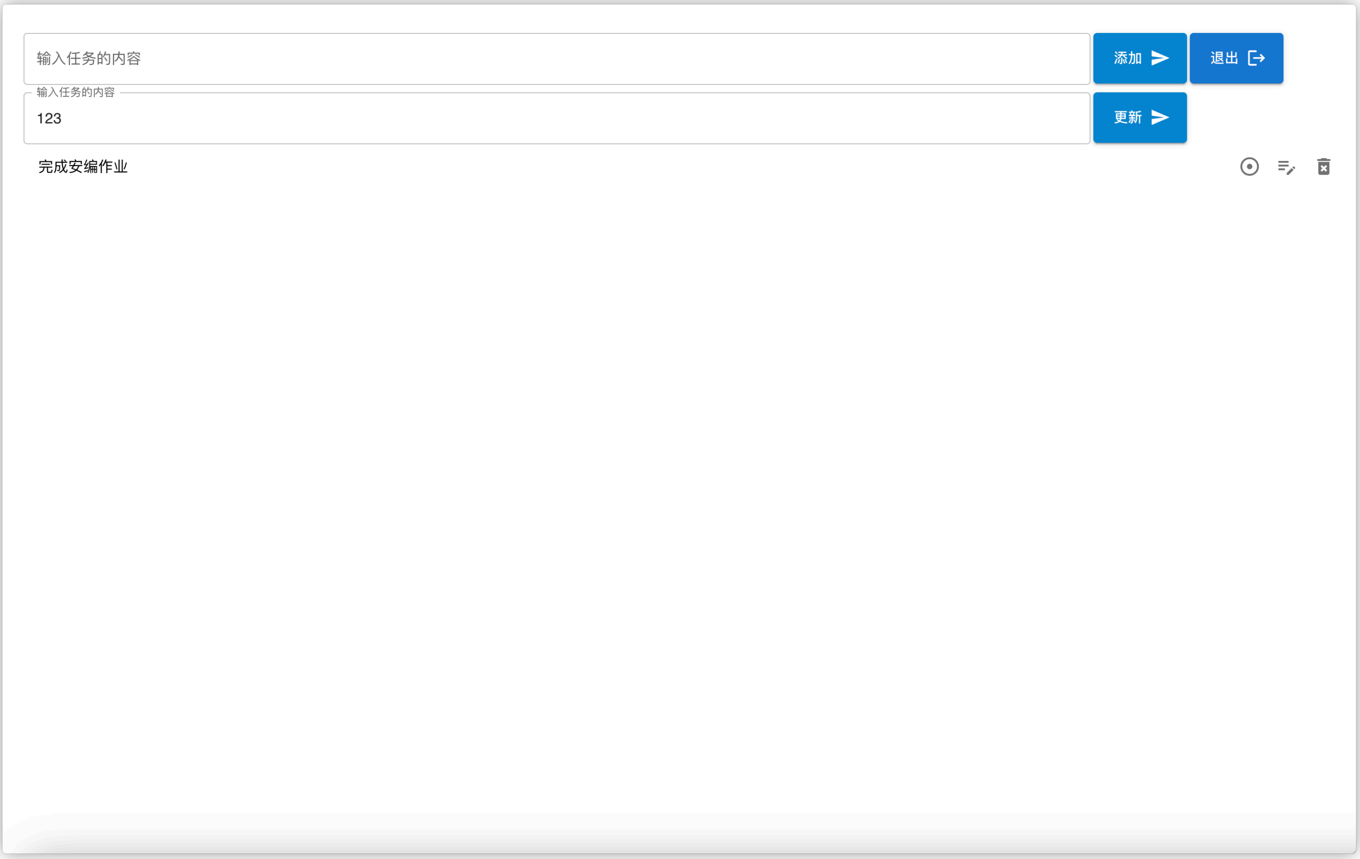
完成安编作业

后端代码：

```java
@PostMapping("/update")
public String update(@RequestBody TodoListPojo todoListPojo, HttpServletResponse
response) {
    try {
        int userId = getUserId();
        todoListPojo.setUserId(userId);
        response.setStatus(HttpServletResponse.SC_OK);
        todoListService.update(todoListPojo);
        return "success";
    } catch (Exception e) {
        e.printStackTrace();
        response.setStatus(HttpServletResponse.SC_BAD_REQUEST);
        return e.toString();
    }
}
```

2.2.5 项目描述编辑

点击第二个按钮，可以对事项进行编辑，点击更新可以更新事项描述。



后端代码：

```java
@PostMapping("/update")
public String update(@RequestBody TodoListPojo todoListPojo, HttpServletResponse response) {
    try {
        int userId = getUserId();
        todoListPojo.setUserId(userId);
        response.setStatus(HttpServletResponse.SC_OK);
        todoListService.update(todoListPojo);
        return "success";
    } catch (Exception e) {
        e.printStackTrace();
        response.setStatus(HttpServletResponse.SC_BAD_REQUEST);
        return e.toString();
    }
}
```

2.2.6 删除事项

　　点击最后一个按钮，可以删除事项。

后端代码：

```java
@PostMapping("/delete")
```

```java
public String delete(@RequestBody TodoListPojo todoListPojo, HttpServletResponse response) {
    try {
        int userId = getUserId();
        if (userId ≠ todoListService.getTodo(todoListPojo.getId()).getUserId()) {
            response.setStatus(HttpServletResponse.SC_UNAUTHORIZED);
            return "unauthorized";
        }
        todoListPojo.setUserId(userId);
        response.setStatus(HttpServletResponse.SC_OK);
        todoListService.delete(todoListPojo.getId());
        return "success";
    } catch (Exception e) {
        e.printStackTrace();
        response.setStatus(HttpServletResponse.SC_BAD_REQUEST);
        return e.toString();
    }
}
```

2.3 前端实现

项目使用react进行开发，使用react-router-dom管理路由，首先在App.js中配置路由，这里只配置了登录、注册和管理待办事项三个路由。

```jsx
import logo from './logo.svg';
import './App.css';
import {
  BrowserRouter as Router,
  Switch,
  Routes,
  Route,
  useHistory,
  useLocation
} from "react-router-dom";
import Login from './pages/Login';
import DashBoard from './pages/DashBoard';
import Register from './pages/Register';
function App() {
  return (
    <Router>
      <div>
        <Routes>
          <Route path="/login" element={<Login />} />
          <Route path="/" element={<Register />} />
          <Route path="/register" element={<Register />} />
          <Route path="/dashboard" element={<DashBoard />} />
        </Routes>
      </div>
    </Router>
  );
}

export default App;

```

注册界面的代码如下，用户输入信息完毕之后，点击注册按钮，浏览器会发送一个 post 请求到后端，后端会处理这个请求，如果注册成功，前端会重定向到登录页面，如果失败，会弹出失败原因。

```
import * as React from 'react';
import Button from '@mui/material/Button';
import CssBaseline from '@mui/material/CssBaseline';
import TextField from '@mui/material/TextField';
import Box from '@mui/material/Box';
import Typography from '@mui/material/Typography';
import Container from '@mui/material/Container';
import axios from 'axios';
import { useStorage } from '../utils/LocalStorageUtil'
import { useNavigate } from 'react-router-dom'
export default function SignIn() {
    const store = useStorage();
    const navigate = useNavigate();
    const handleSubmit = (event) => {
        event.preventDefault();
        const formdata = new FormData(event.currentTarget);
        const data = {
            "username": formdata.get("username"),
            "password": event.currentTarget.get("password")
        }
        axios.post('http://localhost:8080/register', data).then(res => {
            console.log(res);
            navigate("/login")
        }).catch(e => {
            console.log(e);
            alert(e)
        })
    };

    return (
        <Container component="main" maxWidth="xs">
            <CssBaseline />
            <Box
                sx={{
                    marginTop: 8,
                    display: 'flex',
                    flexDirection: 'column',
                    alignItems: 'center',
                }}
            >
                <Typography component="h1" variant="h5">
                    注册
                </Typography>
                <Box component="form" onSubmit={handleSubmit} noValidate sx={{ mt: 1 }}>
                    <TextField
                        margin="normal"
                        required
                        fullWidth
                        id="username"
                        label="用户名"
                        name="username"
                        autoComplete="username"
                        autoFocus
                    />
                    <TextField
                        margin="normal"
                        required
                        fullWidth
                        name="password"
                        label="密码"
                        type="password"
                        id="password"
                        autoComplete="current-password"
                    />
                    <Button
                        type="submit"
                        fullWidth
                        variant="contained"
                        sx={{ mt: 3, mb: 2 }}
                    >
                        注册
                    </Button>
                </Box>
            </Box>
        </Container>
    );
```

```
}
```

登录界面代码如下，首先用户可以输入自己的用户名和密码，然后点击登录，如果登录失败，会弹出失败原因；如果登录成功，那么后端会返回一个jwt token，前端会将这个 token 存储在浏览器的本地存储里面，作为接下来鉴权的媒介，然后跳转到待办事项管理界面。

```
import * as React from 'react';
import Avatar from '@mui/material/Avatar';
import Button from '@mui/material/Button';
import CssBaseline from '@mui/material/CssBaseline';
import TextField from '@mui/material/TextField';
import FormControlLabel from '@mui/material/FormControlLabel';
import Link from '@mui/material/Link';
import Grid from '@mui/material/Grid';
import Box from '@mui/material/Box';
import LockOutlinedIcon from '@mui/icons-material/LockOutlined';
import Typography from '@mui/material/Typography';
import Container from '@mui/material/Container';
import axios from 'axios';
import { useStorage } from '../utils/LocalStorageUtil'
import { useNavigate } from 'react-router-dom'
export default function SignIn() {
    const store = useStorage();
    const navigate = useNavigate();
    const handleSubmit = (event) => {
        event.preventDefault();
        const formdata = new FormData(event.currentTarget);
        const data = {
            "username": formdata.get("username"),
            "password": formdata.get("password")
        }
        axios.post('http://localhost:8080/login', data).then(res => {
            console.log(res);
            store.save("token", res.data);
            navigate("/dashboard")
        }).catch(e => {
            console.log(e);
            alert(e)
        })
    };

    return (
        <Container component="main" maxWidth="xs">
            <CssBaseline />
            <Box
```

```
                sx={{
                    marginTop: 8,
                    display: 'flex',
                    flexDirection: 'column',
                    alignItems: 'center',
                }}
            >
                <Typography component="h1" variant="h5">
                    登录
                </Typography>
                <Box component="form" onSubmit={handleSubmit} noValidate sx={{ mt: 1 }}>
                    <TextField
                        margin="normal"
                        required
                        fullWidth
                        id="username"
                        label="用户名"
                        name="username"
                        autoComplete="username"
                        autoFocus
                    />
                    <TextField
                        margin="normal"
                        required
                        fullWidth
                        name="password"
                        label="密码"
                        type="password"
                        id="password"
                        autoComplete="current-password"
                    />
                    <Button
                        type="submit"
                        fullWidth
                        variant="contained"
                        sx={{ mt: 3, mb: 2 }}
                    >
                        登录
                    </Button>
                    <Grid container>

                        <Grid item>
                            <Link href="/register" variant="body2">
                                {"还没有账号? 请您注册"}
                            </Link>
                        </Grid>
                    </Grid>
                </Box>
            </Box>
        </Container>
    );
}
```

待办事项管理界面的代码如下，首先是一个输入框，在这里可以进行新事项的添加，然后是一个退出登录的按钮，点击这个按钮可以退出当前账户，接下来是用户待办事项的列表，在这里用户可以对自己的待办事项进行管理。

```
import { Box, Button, ListItem, ListItemSecondaryAction, ListItemText, TextField } from '@mui/material';
import SendIcon from '@mui/icons-material/Send';
import axios from 'axios';
import { useStorage } from '../utils/LocalStorageUtil';
import { useNavigate } from 'react-router-dom';
import { useEffect, useState } from 'react';
import LogoutIcon from '@mui/icons-material/Logout';
import { List } from '@mui/material';
import { IconButton } from '@mui/material';
import DeleteForeverIcon from '@mui/icons-material/DeleteForever';
import EditNoteIcon from '@mui/icons-material/EditNote';
import AdjustIcon from '@mui/icons-material/Adjust';
function Iteminput(props) {
    const { refresh, storage } = props;
    const navigate = useNavigate();
    const [content, setContent] = useState("");
    useEffect(() => {
        if (!storage.get("token")) {
            console.log("not login")
            navigate("/login");
        }
    }, [])
    const handleSubmit = () => {
        const data = {
            "content": content
        };
        axios.post('http://localhost:8080/todo/insert', data, {
            headers: {
                "Content-Type": "application/json",
                "Authorization": storage.get("token")
            }
        }).then(res => {
            console.log(res);
            refresh();
        }).catch(e => {
            console.log(e);
            alert(e)
        })
    };
    return (
        <>
            <TextField
                id="outlined-basic"
                label="输入任务的内容"
                variant="outlined"
                value={content}
                onChange={(e) => setContent(e.target.value)}
                sx={{
                    width: '80%'
                }}
            />
            <Button
                variant="contained"
                color="info"
                size="large"
                endIcon={<SendIcon />}
                onClick={handleSubmit}
                sx={{
                    height: '3.4rem',
                    marginLeft: '0.2rem'
                }}
            >
                添加
            </Button>
        </>
    );
}
const MyListItem = (props) => {
    const { item, refresh, storage } = props;
    const [isEditing, setIsEditing] = useState(false);
    const [newContent, setNewContent] = useState(item.content);
    const adjust = () => {
        const data = {
            "id": item.id,
            "finish": !item.finish,
            "content": item.content,
        }
        axios.post('http://localhost:8080/todo/update', data, {
            headers: {
                "Content-Type": "application/json",
                "Authorization": storage.get("token")
            }
        }).then(res ⇒ {
            console.log(res);
            refresh();
        }).catch(e ⇒ {
            console.log(e);
            alert(e)
        })
    }
    const edit = () ⇒ {
        setIsEditing(true);
    }
    const deleteTodo = () ⇒ {
        const data = {
            "id": item.id,
        }
        axios.post('http://localhost:8080/todo/delete', data, {
            headers: {
                "Content-Type": "application/json",
                "Authorization": storage.get("token")
            }
        }).then(res ⇒ {
            console.log(res);
            refresh();
        }).catch(e ⇒ {
            console.log(e);
            alert(e)
        })
    }
    const handleSubmit = () ⇒ {
        const data = {
            "id": item.id,
            "finish": item.finish,
            "content": newContent,
        }
        axios.post('http://localhost:8080/todo/update', data, {
            headers: {
                "Content-Type": "application/json",
                "Authorization": storage.get("token")
            }
        }).then(res ⇒ {
            console.log(res);
            setIsEditing(false);
            refresh();
        }).catch(e => {
            console.log(e);
            alert(e)
        })
    }
    return (isEditing ? <Box>
        <TextField
            id="outlined-basic"
            label="输入任务的内容"
            variant="outlined"
            value={newContent}
            onChange={(e) => setNewContent(e.target.value)}
            sx={{
                width: '80%'
            }}
        />
        <Button
            variant="contained"
            color="info"
            size="large"
            endIcon={<SendIcon />}
            onClick={handleSubmit}
            sx={{
                height: '3.4rem',
                marginLeft: '0.2rem'
            }}
        >
            更新
        </Button>
    </Box> :
        <>
            <ListItem>
                <ListItemText sx={{
                    textDecorationLine: item.finish ? 'line-through' : 'none'
                }}>
                    {item.content}
                </ListItemText>
                <ListItemSecondaryAction>
                    <IconButton aria-label="edit" onClick={adjust}>
                        <AdjustIcon />
                    </IconButton>
                    <IconButton aria-label="edit" onClick={edit}>
```

```
                        <EditNoteIcon />
                    </IconButton>
                    <IconButton aria-label="delete" onClick={deleteTodo}>
                        <DeleteForeverIcon />
                    </IconButton>
                </ListItemSecondaryAction>
            </ListItem >

        </>
    )
}
const DashBoard = () => {
    const itemList = [];
    const storage = useStorage();
    const navigate = useNavigate();
    let [items, setItems] = useState([]);
    useEffect(() => {
        const fetchData = async () => {
            await refresh();
        }
        fetchData();
    }, [])
    const refresh = async () => {
        console.log(refresh)
        await axios.post('http://localhost:8080/todo/all', {}, {
            headers: {
                "Content-Type": "application/json",
                "Authorization": storage.get("token")
            }
        }).then((res) => {
            console.log(res);
            setItems(res.data);
        })
    }
    const logout = () => {
        storage.clear();
        navigate("/login")
    }
    return (
        <Box sx={{
            marginTop: '2rem',
            marginLeft: '2rem'
        }}>
            <Iteminput refresh={refresh} storage={storage} />
            <Button
```

```
                    variant="contained"
                    color="primary"
                    size="large"
                    endIcon={<LogoutIcon />}
                    onClick={logout}
                    sx={{
                        height: '3.4rem',
                        marginLeft: '0.2rem'
                    }}
                >
                    退出
                </Button>
                <List>
                    {
                        items.map((item) => {
                            return (
                                <MyListItem
                                    key={item.id}
                                    item={item}
                                    refresh={refresh}
                                    storage={storage}
                                />
                            );
                        })
                    }
                </List>
            </Box>
        )
}
export default DashBoard;
```

2.4 部署和测试

- 前端部署
    - 在部署前端之前，请您确认自己的计算机上具有nodejs16及以上版本，以及yarn程序。
    - 请您在命令行中进入frontend文件夹，然后执行 `yarn install` 命令，之后执行 `yarn start` 即可，如果启动失败，请您检查端口 `3000` 是否已经占用。
- 后端部署
    - 请您确保计算机上 `JAVA` 版本为 `java18` 及以上，确保计算机安装有 `maven`，mysql8.0以上。
    - 之后，请您修改application.properties文件中的spring.datasource.password为您的 `mysql` 密码，并且在本地创建一个名为sec 的数据库，在里面执行init.sql中的命令。
    - 之后，请您在项目目录下执行`mvn spring-boot:run`命令，等待项目启动即可。

3. 项目演示

- 项目的演示视频，请参见 `https://www.aliyundrive.com/s/4R29aFGm22h`。