

華東理工大學

信息科学与工程学院

《软件工程》 实验报告

系 别 计算机系

专 业 计算机科学与技术

年 级 2020 级

姓 名 刘子言

指导教师 阮 彤

2022-2023 学年 第 1 学期

实验四 软件架构

一、实验目的

- 1、熟悉前后端分离下的软件架构；
- 2、实现前后端通信；
- 3、实现后端与数据库通信；
- 4、完成前后端分离实验 Todos。

二、实验装置

IDE: VS Code

后端 (Java): jdk8, jre8

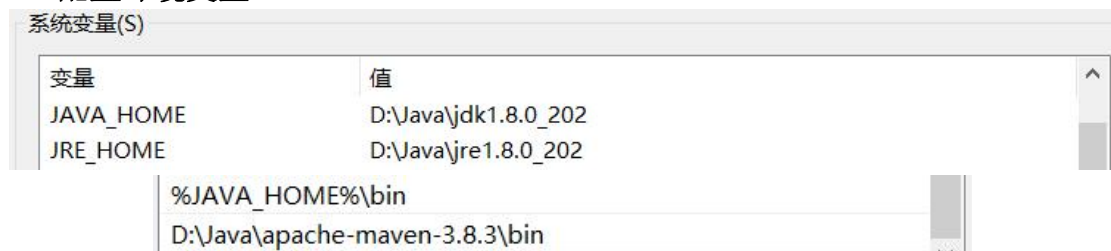
后端 (node): node, npm

数据库: MySQL, navicat

前端: npm

三、实验内容

1、配置环境变量

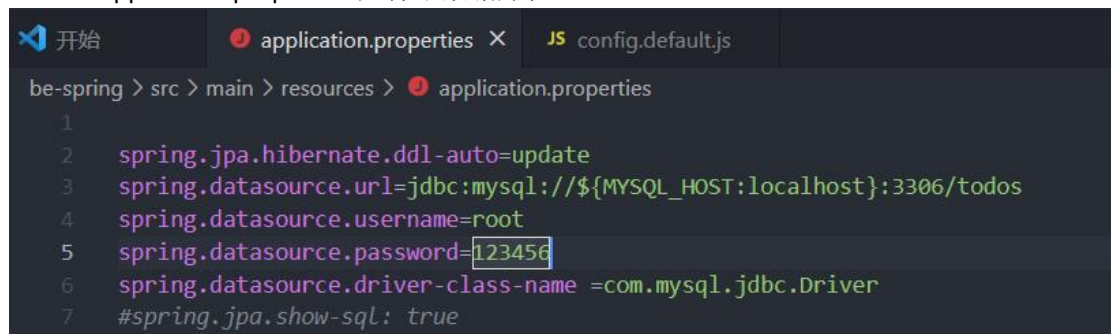


2、连接数据库

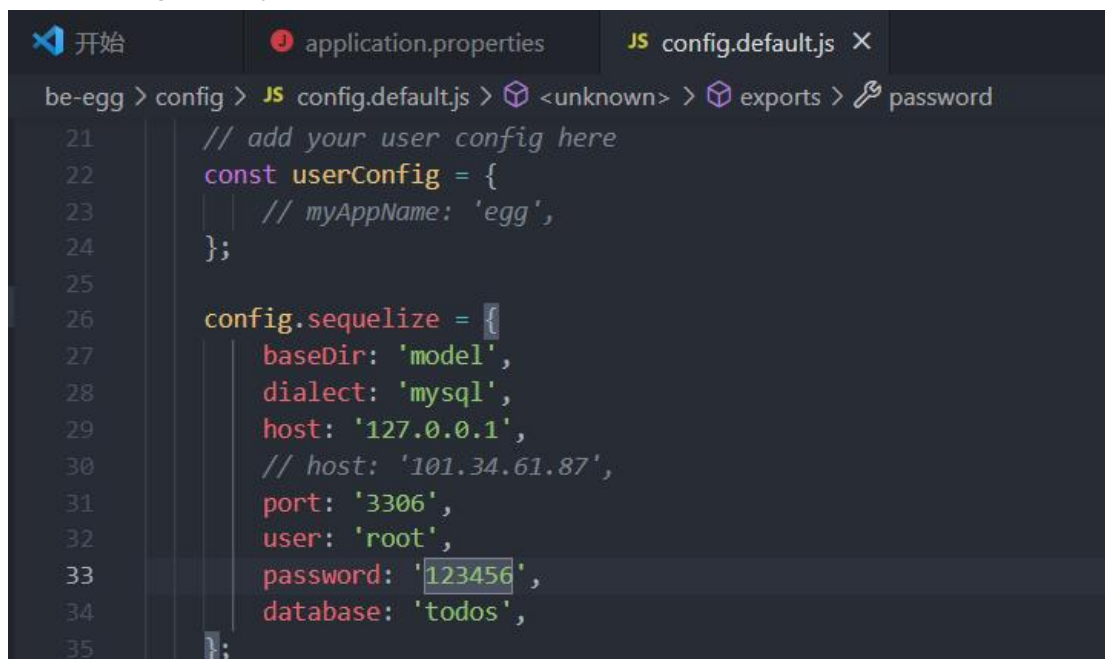
- 启动 mysql 数据库



- 配置 application.properties 文件中数据库信息



- 配置 config.default.js 文件中数据库信息

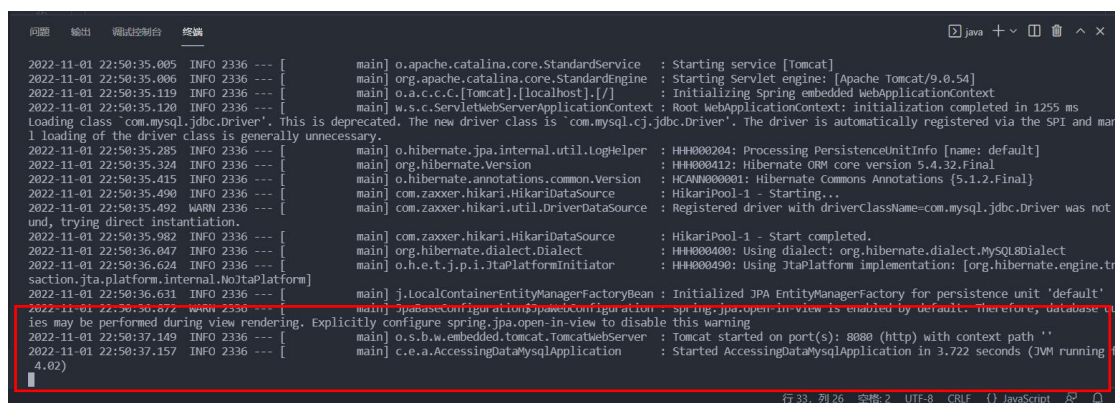


```
be-egg > config > JS config.default.js > <unknown> > exports > password
21 // add your user config here
22 const userConfig = {
23   // myAppName: 'egg',
24 };
25
26 config.sequelize = {
27   baseDir: 'model',
28   dialect: 'mysql',
29   host: '127.0.0.1',
30   // host: '101.34.61.87',
31   port: '3306',
32   user: 'root',
33   password: '123456',
34   database: 'todos',
35 };
```

3、启动项目前后端

- 启动后端

在...\restfulDemo\be-spring 目录下执行 mvn spring-boot:run 命令:

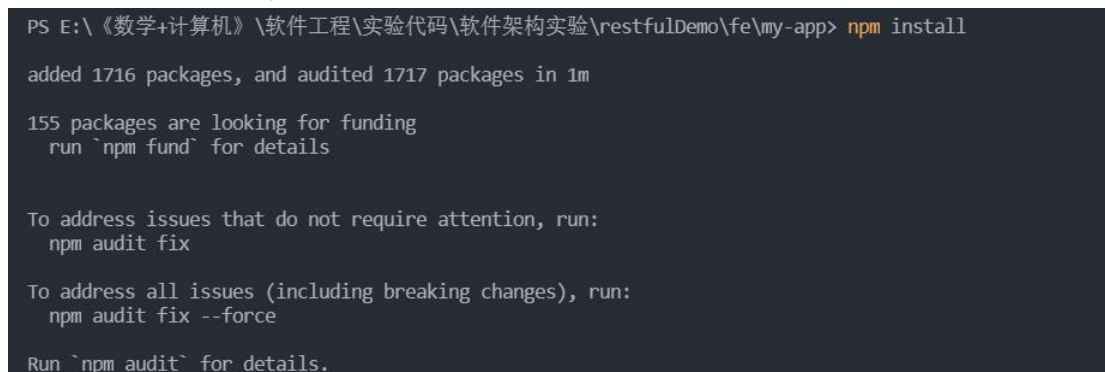


```
2022-11-01 22:50:35.005 INFO 2336 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2022-11-01 22:50:35.006 INFO 2336 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.54]
2022-11-01 22:50:35.119 INFO 2336 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2022-11-01 22:50:35.120 INFO 2336 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 1255 ms
Loading class 'com.mysql.jdbc.Driver'. This is deprecated. The new driver class is 'com.mysql.cj.jdbc.Driver'. The driver is automatically registered via the SPI and manual loading of the driver class is generally unnecessary.
2022-11-01 22:50:35.285 INFO 2336 --- [main] o.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing PersistenceUnitInfo [name: default]
2022-11-01 22:50:35.324 INFO 2336 --- [main] org.hibernate.Version : HHH000412: Hibernate ORM core version 5.4.32.Final
2022-11-01 22:50:35.415 INFO 2336 --- [main] o.hibernate.annotations.common.Version : HCANN000001: Hibernate Commons Annotations {5.1.2.Final}
2022-11-01 22:50:35.490 INFO 2336 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2022-11-01 22:50:35.492 WARN 2336 --- [main] com.zaxxer.hikari.util.DriverDataSource : Registered driver with driverClassName=com.mysql.jdbc.Driver was not found, trying direct instantiation.
2022-11-01 22:50:35.902 INFO 2336 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2022-11-01 22:50:36.047 INFO 2336 --- [main] org.hibernate.dialect.Dialect : HHH000400: Using dialect: org.hibernate.dialect.MySQL8Dialect
2022-11-01 22:50:36.624 INFO 2336 --- [main] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
2022-11-01 22:50:36.631 INFO 2336 --- [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2022-11-01 22:50:36.872 WARN 2336 --- [main] org.springframework.orm.jpa.AbstractEntityManagerFactoryBean : spring.jpa.open-in-view is enabled by default. Therefore, database queries may be performed during view rendering. Explicitly configure spring.jpa.open-in-view to disable this warning
2022-11-01 22:50:37.149 INFO 2336 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2022-11-01 22:50:37.157 INFO 2336 --- [main] c.e.a.AccessingDataMysqlApplication : Started AccessingDataMysqlApplication in 3.722 seconds (JVM running for 4.02s)
```

后端启动成功。

- 启动前端

在...\restfulDemo\fe\my-app 目录下执行 npm install, 随后执行 npm start(由于版本原因, 我未在本实验中使用 yarn):



```
PS E:\《数学+计算机》\软件工程\实验代码\软件架构实验\restfulDemo\fe\my-app> npm install
added 1716 packages, and audited 1717 packages in 1m
155 packages are looking for funding
run `npm fund` for details
To address issues that do not require attention, run:
  npm audit fix
To address all issues (including breaking changes), run:
  npm audit fix --force
Run `npm audit` for details.
```

```
PS E:\《数学+计算机》\软件工程\实验代码\软件架构实验\restfulDemo\fe\my-app> npm start

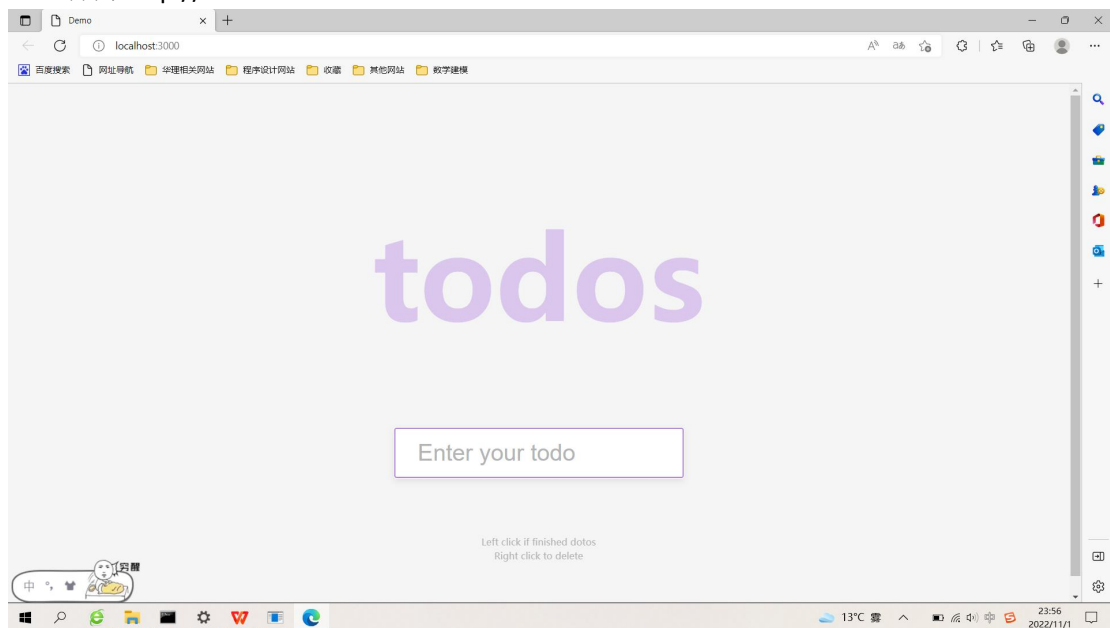
Compiled successfully!

You can now view my-app in the browser.

Local:      http://localhost:3000
On Your Network:  http://10.103.106.125:3000

Note that the development build is not optimized.
To create a production build, use npm run build.
```

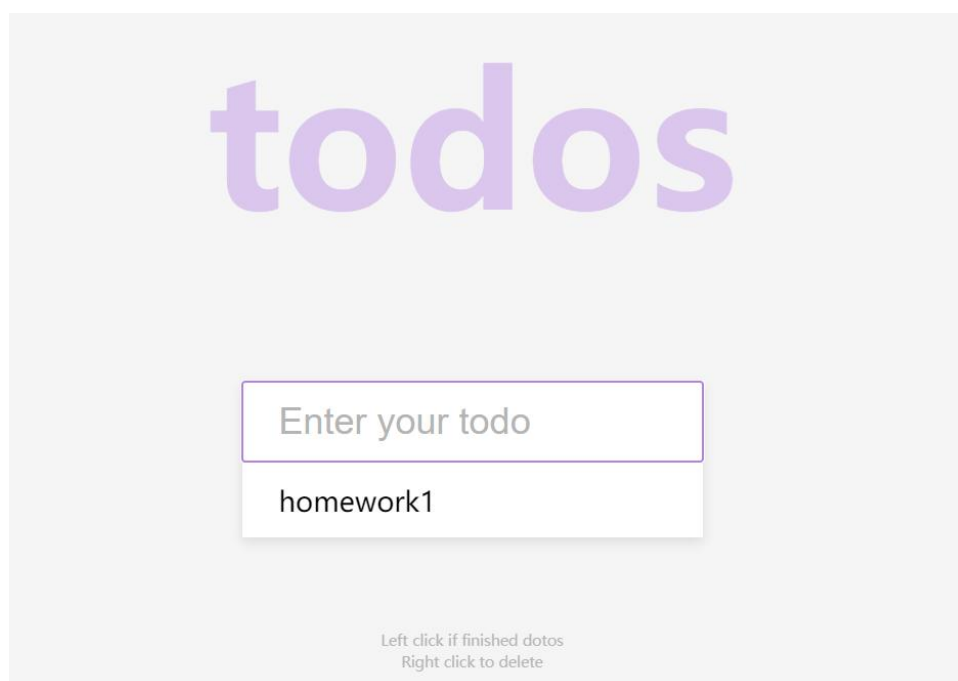
访问 <http://localhost:3000>:



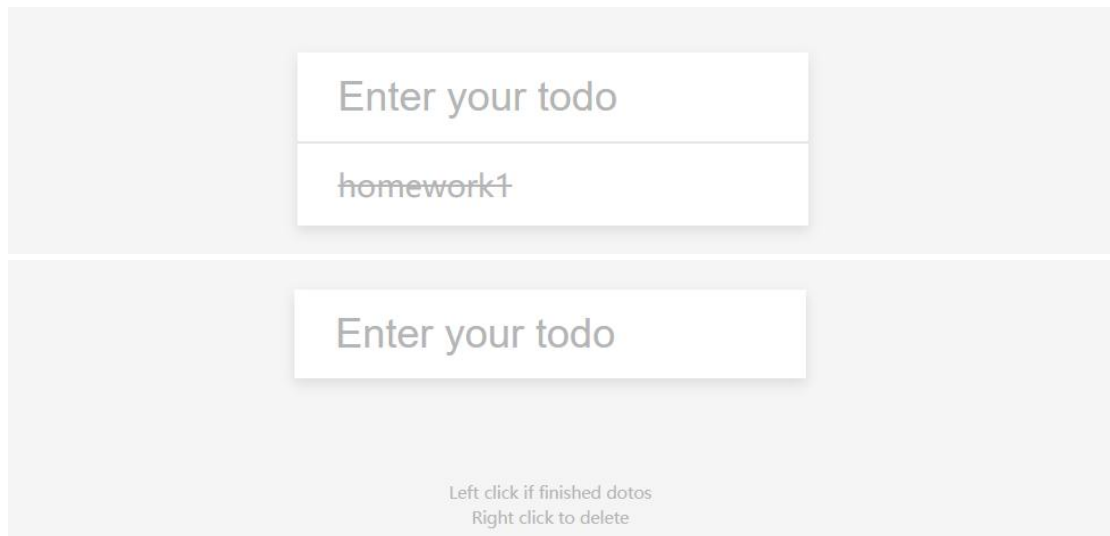
前端启动成功！

4、Todes 实验代码修改

- 现在进入页面，前端会自动向后台发起请求，获取数据库中所有数据
例如：输入 homework1，回车键，则在界面中会显示数据库里所有的数据。



鼠标左击完成该项 **todo** 任务，右击删除该项任务：



- 实验需求：

- 进入页面后不要自动发送请求
- 在页面中添加一个 **button**
- 点击 **button** 后向后台发起请求
- 要获取的内容为，数据库中的数据

- 修改的代码如下：

注释掉自动向后端发送请求获取数据和更新数据的函数：

```
25 // component did mount 自动获取后台数据
26 // useEffect(()=>{
27 //   axios.get(`${useApi}/getTodos`).then((res) => {
28 //     const data = res.data
29 //     console.log(`data from ${useApi}`, data)
30 //     setLocalTodos(data)
31 //   }).catch((err) => {
32 //     window.alert('获取后台数据失败')
33 //   })
34 // }, [])
35
36 // state update 自动更新后台数据
37 // useEffect(() => {
38 //   localStorage.setItem('todos', JSON.stringify(localTodos))
39 // }, [localTodos])
```

前端页面展示的代码修改如下：

```

154     return (
155       <div className="App">
156         <h1>todos</h1>
157         <button onClick={getTodos} type="button" class="button green">Show all todeos</button>
158         <form id="form" onSubmit = {e => addTodo(e)}>
159           <input id="input" ref={input} type="text" placeholder="Enter your todo" />
160           <ul id="todos">
161             {
162               localTodos.map(item => {
163                 return (
164                   <li
165                     className = {item.finished ? "todo finished" : "todo"}
166                     key = {item.id}
167                     onClick = {(e) => finishTodo(item)}
168                     onContextMenu = {e => delTodo(e, item.id)}>{item.text}</li>
169                 )
170               })
171             }
172           <li>数据库中共{num}条数据</li>
173         </ul>
174       </form>
175       <small>
176         Left click if finished dots
177         <br/>
178         Right click to delete
179       </small>
180     </div>
181   );

```

点击 button 后调用的函数如下“getTodos”:

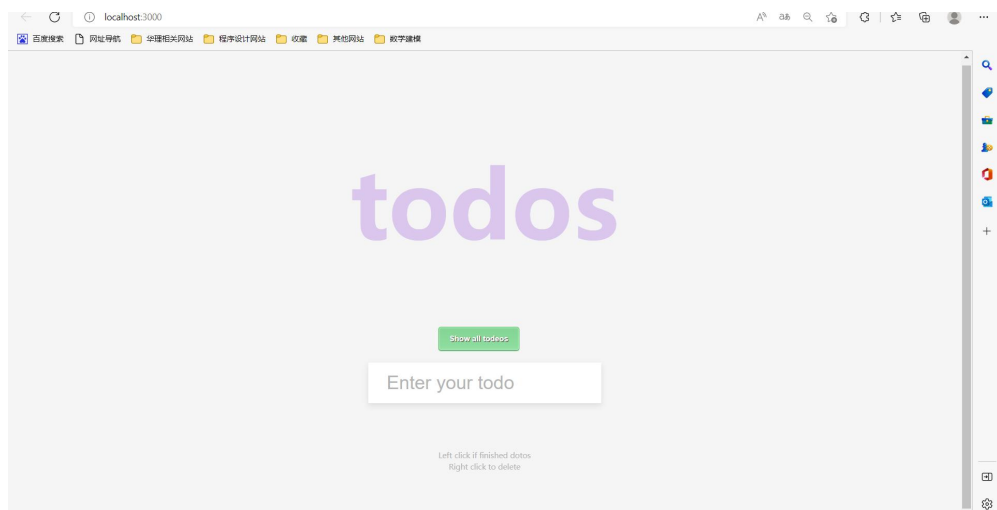
```

const input = useRef()
// const [localTodos, setLocalTodos] = useState(localStorage.getItem('todos') ? JSON.parse
const [localTodos, setLocalTodos] = useState([]) // null -> []
const [clickFreeze, setClickFreeze] = useState(false)

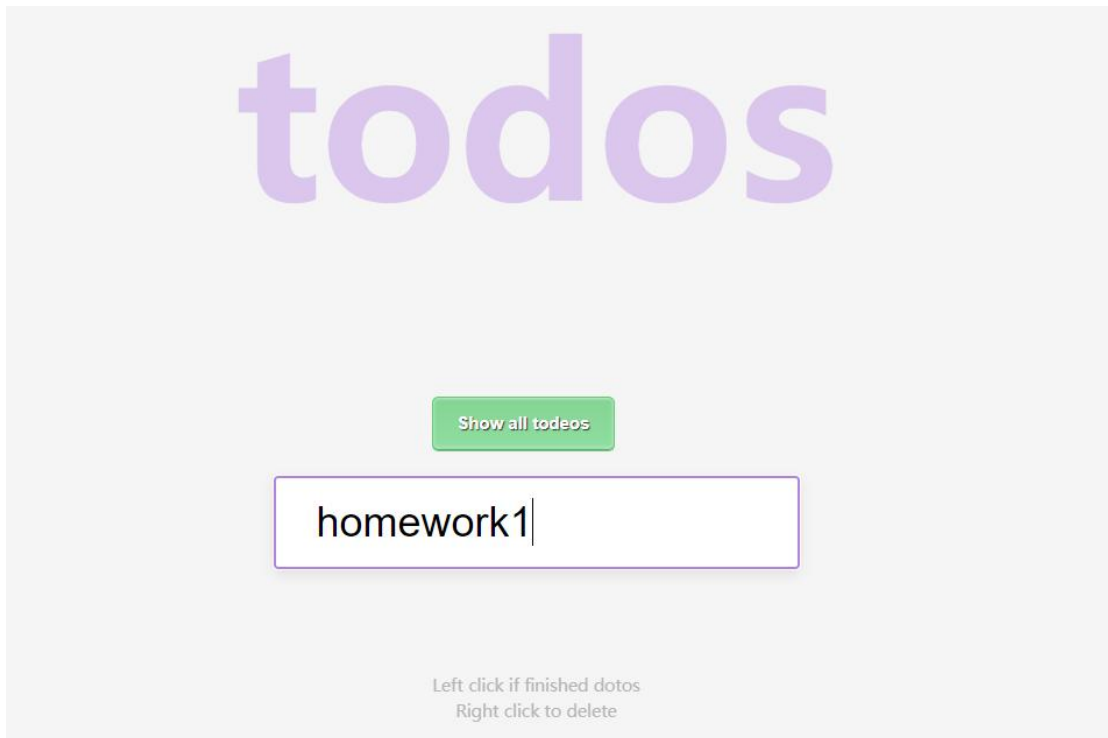
// 点击按钮以后, 获取后台数据
const getTodos=()=>{
  axios.get(`/${useApi}/getTodos`).then((res) => {
    const data = res.data
    //console.log(`data from ${useApi}`, data)
    setLocalTodos(data)
    window.alert('获取后台数据库成功')
  }).catch((err) => {
    window.alert('获取后台数据失败')
  })
}

```

• 演示结果如下:

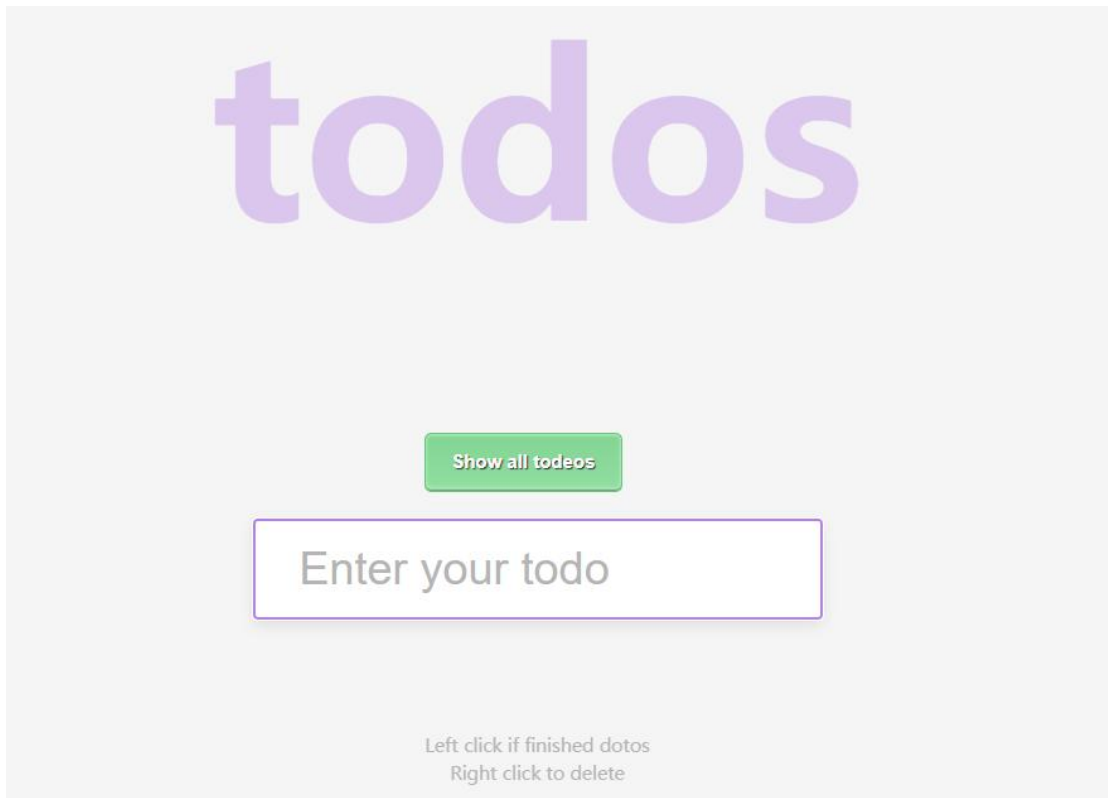


输入一条 homework1:



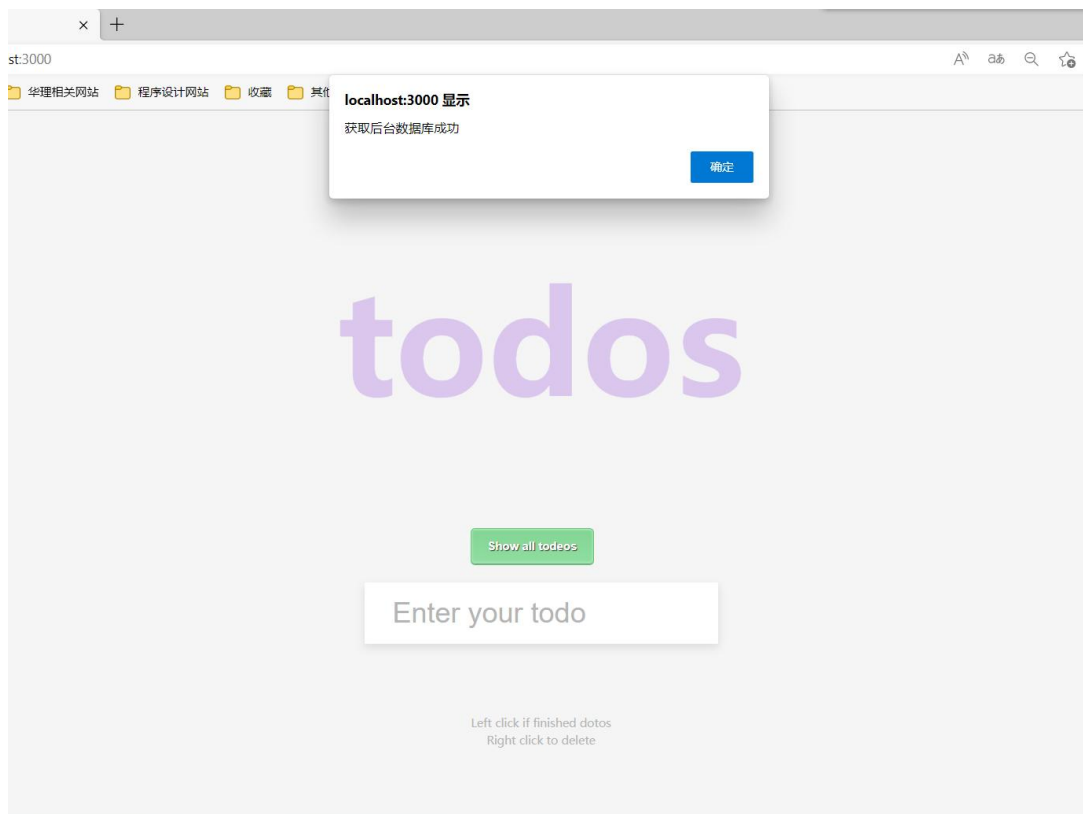
The screenshot shows the 'todos' application interface. At the top, the word 'todos' is displayed in a large, light purple font. Below it is a green button labeled 'Show all todos'. Underneath the button is a text input field with a purple border, containing the text 'homework1' and a cursor at the end. At the bottom, there is a small text label that reads 'Left click if finished todos' and 'Right click to delete'.

回车提交，只提交数据到数据库，不会自动向后台发送请求显示所有数据：

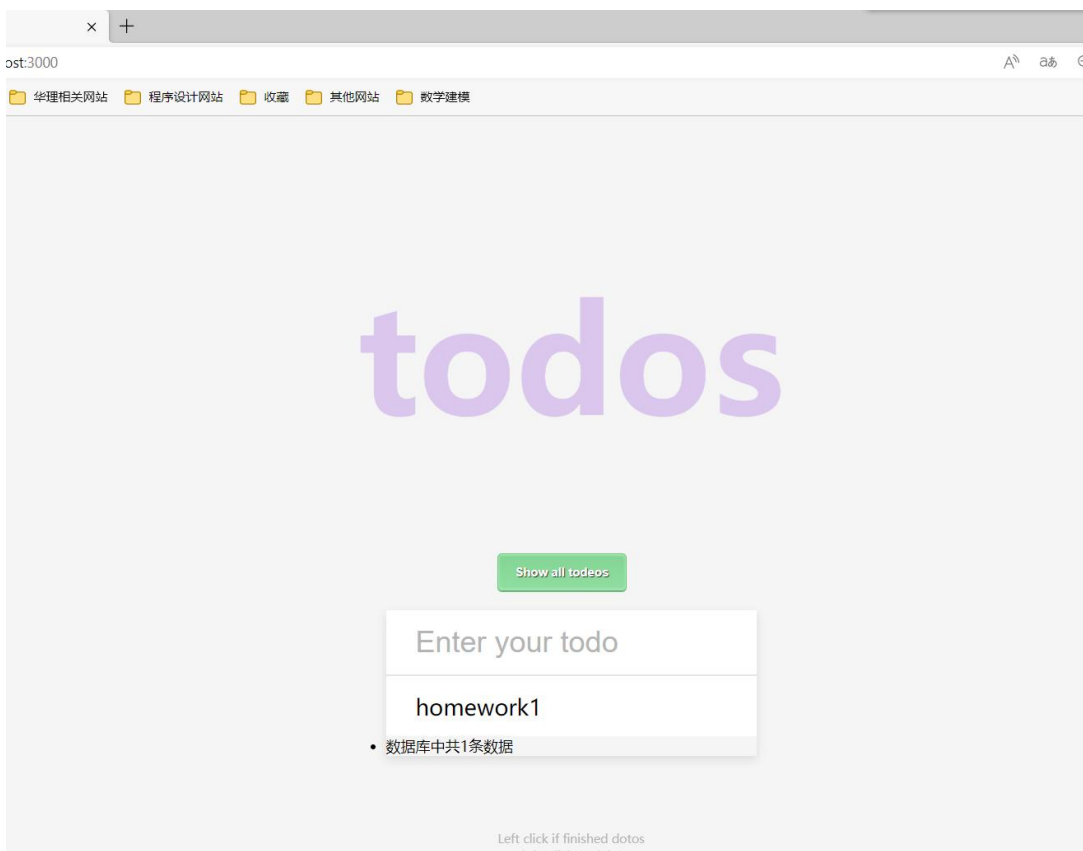


The screenshot shows the 'todos' application interface. At the top, the word 'todos' is displayed in a large, light purple font. Below it is a green button labeled 'Show all todos'. Underneath the button is a text input field with a purple border, containing the placeholder text 'Enter your todo'. At the bottom, there is a small text label that reads 'Left click if finished todos' and 'Right click to delete'.

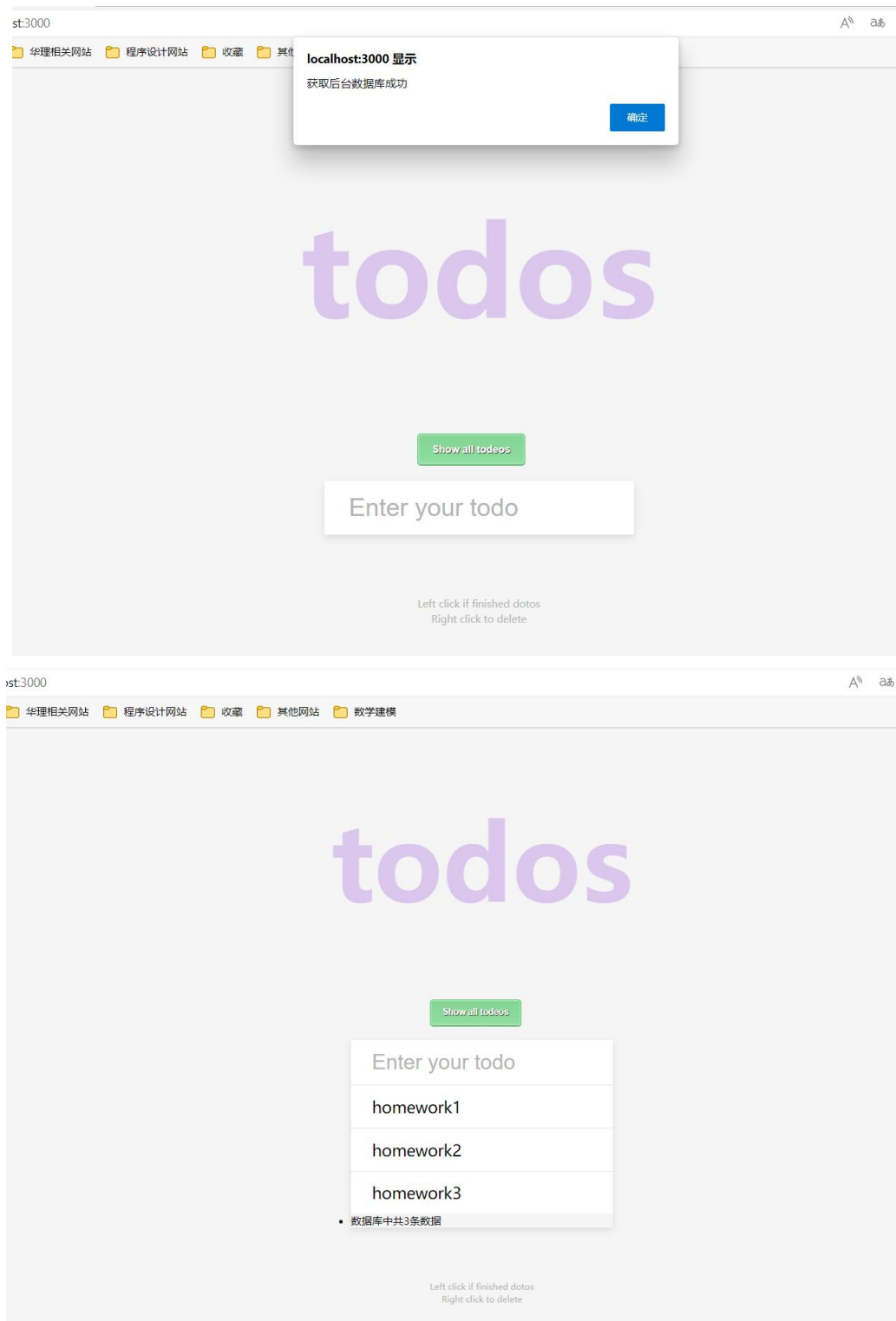
点击 **button**，后向后台发起请求：



再点击确定按钮，即可显示后台数据信息：



再多添加几条 todos 信息，点击 button 按钮，界面结果如下：



四、实验心得

通过本实验，我基本了解了基于 node+java+mysql 前后端分离下的软件架构，掌握了前后端通信及其实现，并在 Todos 实操实验中学习到了前后端交互的技能，希望通过后期再进行自我巩固，能够丰富对软件架构的认识，并将理论运用在项目实践中。