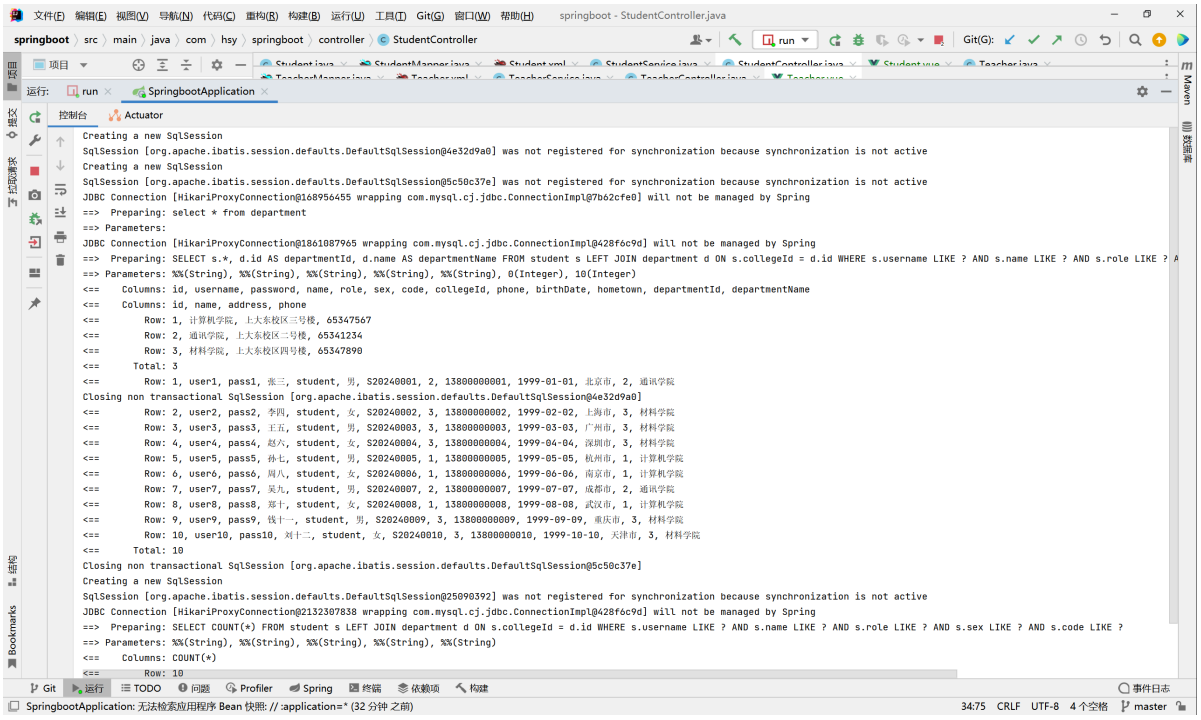
前后端具体返回信息

访问地址 http://localhost:8080/student

后端：



从输出日志中，我们可以直观地看到 MyBatis 执行过程中的几个核心步骤：从连接数据库、执行 SQL 查询、到最终返回数据的全过程。下面我们结合你的日志和 `findAllStudentsWithDepartment` 函数的逻辑，详细讲解访问数据库的流程。

---

1. 创建 SqlSession

日志中多次提到：

```
Creating a new SqlSession
SqlSession [org.apache.ibatis.session.defaults.DefaultSqlSession@...] was not registered for synchronization because synchronization is not active
```

- **含义**：`SqlSession` 是 MyBatis 和数据库交互的关键组件，它封装了所有与数据库操作相关的逻辑，比如查询、插入、更新等。
- **实现原理**：Spring 中的 `Mapper` 接口被 `MyBatis` 自动代理，通过 `SqlSession` 来管理与数据库的交互。每个操作都会开启一个 `SqlSession`。

---

2. 获取 JDBC Connection

日志中的：

```
JDBC Connection [HikariProxyConnection@...] will not be managed by Spring
```

- **含义**：`Hikari` 是常用的数据库连接池，MyBatis 使用连接池来高效地管理数据库连接。这里显示的是 MyBatis 从 `Hikari` 连接池中获取的数据库连接。
- **实现原理**：数据库连接池可以复用连接，避免频繁创建/关闭连接的开销，从而提升性能。

---

3. 准备和执行 SQL 查询

以下是你的查询逻辑对应的 SQL 准备和执行：

```
==>  Preparing: SELECT s.*, d.id AS departmentId, d.name AS departmentName FROM student s LEFT JOIN department d ON s.collegeId = d.id WHERE s.username LIKE ? AND s.name LIKE ? AND s.role LIKE ? AND s.sex LIKE ? AND s.code LIKE ? LIMIT ?, ?
==> Parameters: %%(String), %%(String), %%(String), %%(String), %%(String), 0(Integer), 10(Integer)
```

SQL 准备阶段

- **SQL 语句**：MyBatis 使用 `@Select` 注解或者 XML 文件中的 SQL 模板，将查询动态参数（如 `?`）进行占位，等待后续传入参数。
- **预编译**：通过 JDBC 的 `PreparedStatement` 机制，SQL 在发送到数据库前会进行一次预编译，以提高安全性和性能。

参数绑定

- `Parameters`

  表示动态参数，具体内容：
  - `%` 是 SQL 通配符，`%%` 代表模糊查询所有数据。
  - `0(Integer)` 和 `10(Integer)` 是分页参数，表示查询第 1 页的 10 条数据。

---

4. 获取查询结果

执行 SQL 查询后，数据库返回结果，MyBatis 处理结果并打印：

```
<==    Columns: id, username, password, name, role, sex, code, collegeId, phone, birthDate, hometown, departmentId, departmentName
<==        Row: 1, user1, pass1, 张三, student, 男, S20240001, 2, 13800000001, 1999-01-01, 北京市, 2, 通讯学院
...
<==      Total: 10
```

- **列信息**：日志显示了查询结果的列，包括 `student` 表的字段（`id`, `username` 等）以及 `department` 表的字段（`departmentId`, `departmentName`）。
- **行数据**：每行数据对应数据库中一条记录，MyBatis 会根据 `@Result` 注解或 XML 文件中的 `resultMap` 将查询结果映射到 Java 对象中。

---

5. 关闭 SqlSession

```
Closing non transactional SqlSession
[org.apache.ibatis.session.defaults.DefaultSqlSession@...]
```

- **含义**：SqlSession 在事务执行完成后会被关闭，并释放相关的资源。
- **实现原理**：MyBatis 在每次操作后都会确保 `SqlSession` 被正确关闭，以避免资源泄漏。这里的非事务模式意味着这次查询没有启动事务。

数据流总结

1. **Controller** 层调用 `findAllStudentsWithDepartment` 方法。
2. **Service** 层调用 `Mapper` 的方法，这里由 MyBatis 代理类接管。
3. **Mapper** 层将 SQL 模板和参数交由 `SqlSession` 处理。
4. **SqlSession**：
    - 从连接池中获取数据库连接。
    - 准备 SQL 并将参数绑定到占位符。
    - 通过 JDBC 执行 SQL，获取结果集。
5. **MyBatis** 将结果集解析为 Java 对象列表（如 `List<Student>`），返回给 Service 层。
6. **Controller** 层将结果通过 HTTP 响应发送给前端。

小结

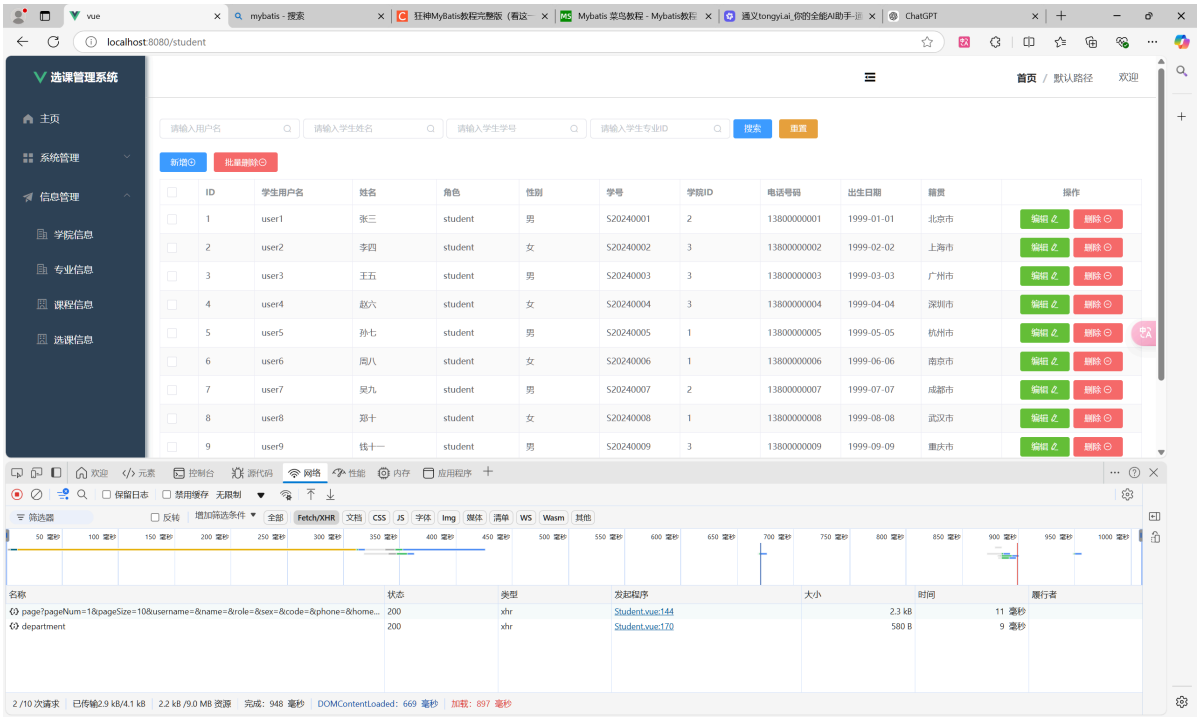
访问数据库的工具核心是 MyBatis：

- **动态 SQL** 提高了查询的灵活性。
- **ORM 映射** 简化了 SQL 结果与 Java 对象的转换。
- **与 Spring 集成** 提供了事务管理和依赖注入，降低了开发难度。

你可以将这整个过程比喻为：

- **Controller** 是客户下订单。
- **Service** 是服务员传递订单。
- **Mapper + SqlSession** 是厨房完成烹饪。
- **数据库** 是食材来源。
- **最终返回** 是把处理好的菜品送到客户面前。

前端：


选课管理系统
主页
系统管理
信息管理
学院信息
专业信息
课程信息
选课信息
首页 / 默认路径
欢迎
请输入用户名
请输入学生姓名
请输入学生学号
请输入学生专业ID
搜索
重置
新增
批量删除
ID
学生用户名
姓名
角色
性别
学号
学院ID
电话号码
出生日期
籍贯
操作
1 user1 张三 student 男 S20240001 2 13800000001 1999-01-01 北京市
2 user2 李四 student 女 S20240002 3 13800000002 1999-02-02 上海市
3 user3 王五 student 男 S20240003 3 13800000003 1999-03-03 广州市
4 user4 赵六 student 女 S20240004 3 13800000004 1999-04-04 深圳市
5 user5 孙七 student 男 S20240005 1 13800000005 1999-05-05 杭州市
6 user6 周八 student 女 S20240006 1 13800000006 1999-06-06 南京市
7 user7 吴九 student 男 S20240007 2 13800000007 1999-07-07 成都市
8 user8 郑十 student 女 S20240008 1 13800000008 1999-08-08 武汉市
9 user9 钱十一 student 男 S20240009 3 13800000009 1999-09-09 重庆市
编辑
删除
名称
状态
类型
发起程序
大小
时间
page?pageNum=1&pageSize=10&username=&name=&role=&sex=&code=&phone=&home...
200
xhr
Student.vue:144
2.3 kB
11 毫秒
department
200
xhr
Student.vue:170
580 B
9 毫秒