成 绩 评 定 表

学生姓名	宁高聪	班级学号	1503130115
专　业	网络工程	课程设计题目	基于 web 的教室管理系统
评 语	组长签字：		
成绩			
日期	2017 年 12 月 22 日		

课程设计任务书

学　院	信息科学与工程学院	专 业	网络工程
学生姓名	宁高聪	班级学号	1503130115
课程设计题目	基于 web 的教室管理系统		

实践教学要求与任务：

1. 任务描述

使用 JAVA 作为编程语言，MySQL 作为数据库，完成基于 Web 的教室管理系统的设计与实现。基于 Web 的教室管理系统包括前台页面设计、后台业务逻辑两个部分，主要功能包括教师信息的管理、教室信息管理、教室安排信息管理等。

2. 具体要求（每个组员根据各自的工作从下面的任务选择或新增相应的工作）

（1）使用 HTML + JavaScript + CSS 设计前台页面。

（2）使用 MySQL 设计数据库。

（3）使用 Hibernate 建立表到对象的映射。

（4）使用 Struts2 控制后台业务逻辑要求。

3. 本人任务

数据库中各个表的设计；

使用 Hibernate 完成对象到表的映射。

工作计划与进度安排：

17 周：

布置课程设计任务，查阅资料，分组设计

实验室编写代码与调试，验收，答辩，编写课程设计报告。

指导教师：	专业负责人：	学院教学副院长：
2017 年 12 月 22 日	2017 年 12 月 22 日	2017 年 12 月 22 日

摘 要

通过调查研究，发现教室管理大多采用人工方式，不便于管理和用户使用。本系统针对教室管理人员和用户，实现数据的增删改，方便用户操作和系统的实现。主要实现了空教室查询，教师调课的管理，设备维护管理和教室借用的管理。空教室查询包括空闲教室查询。借用管理完成教室的借用处理和记录等。

关键词　B/S 架构；Hibernate；Struts2；MySQL

目 录

1 问题描述

随着计算机技术、网络技术和信息技术的发展，现在办公系统更趋于系统化、学化和网络化。网络办公自动化系统是计算机技术和网络迅速发展的一个办公应用解决方案，它的主要目的是实现信息交流和信息共性，提供协同工作的手段，提高办公的效率，让人们从繁琐的有纸办公中解脱出来。现在许多的机关单位的人事管理水平还停留在纸介质的基础上，这样的机制已经不能适应时代的发展，因为它浪费了许多的人力和物力，在信息时代这种传统的管理方法必然被计算机为基础的信息管理所取代。本系统是对教室的使用情况进行管理，为用户提供了一套操作简单、使用可靠、界面友好、易于管理和使用的处理工具。本系统对教室使用情况进行统一处理，避免数据存取、数据处理的重复，提高工作效率，减少了系统数据处理的复杂性。本系统不仅使管理人员从繁重的工作中解脱出来，而且提高了教室管理的效率，提高了教室管理的科学性，方便了用户查询、管理人员进行管理。

2 需求分析

2.1 系统功能结构

本系统采用的是 B/S 模式，其主要的功能在服务器上执行。B/S 结构的客户端可以完成浏览、查询、数据输入等简单功能，绝大部分工作要由服务器承担，包括对数据的保存，如：数据存储、恢复，以及对系统失效的后果及恢复的处理方法等。有教师账号的教师可以通过浏览器访问查看教室的使用情况以及线上申请教室的使用权限等。

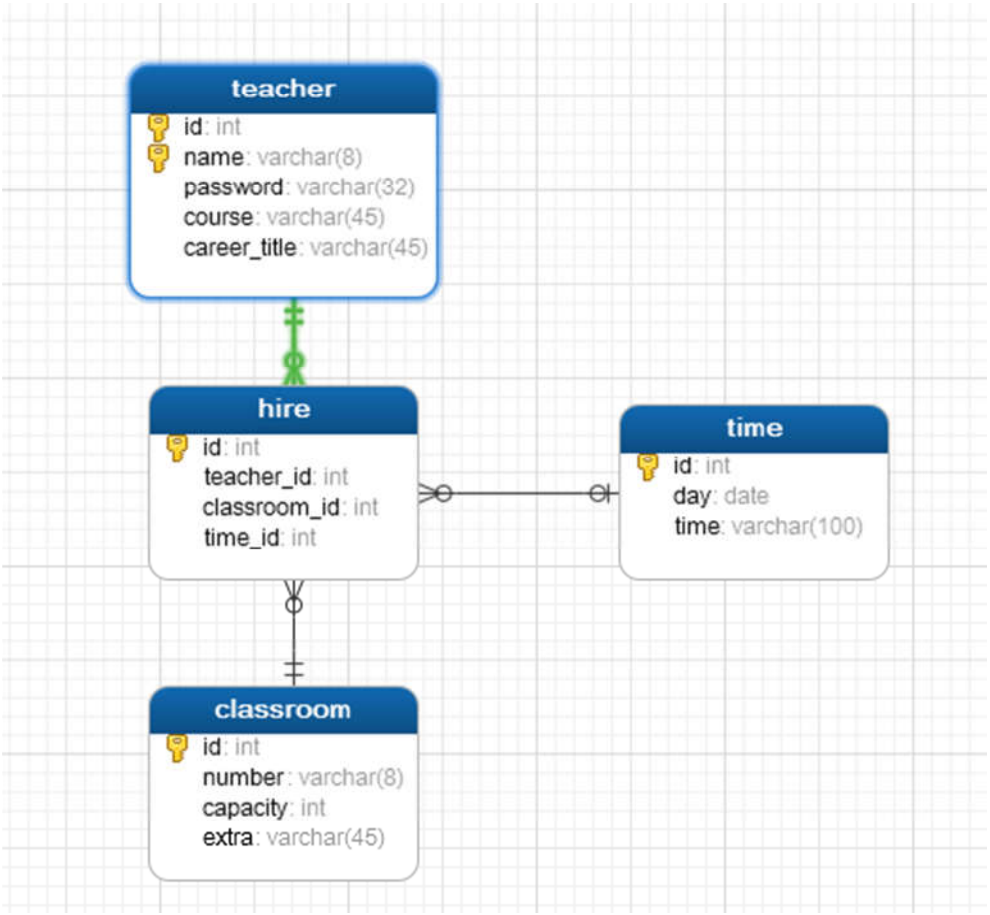

图 2.1 数据库 E-R 模型图

2.2 逻辑结构设计

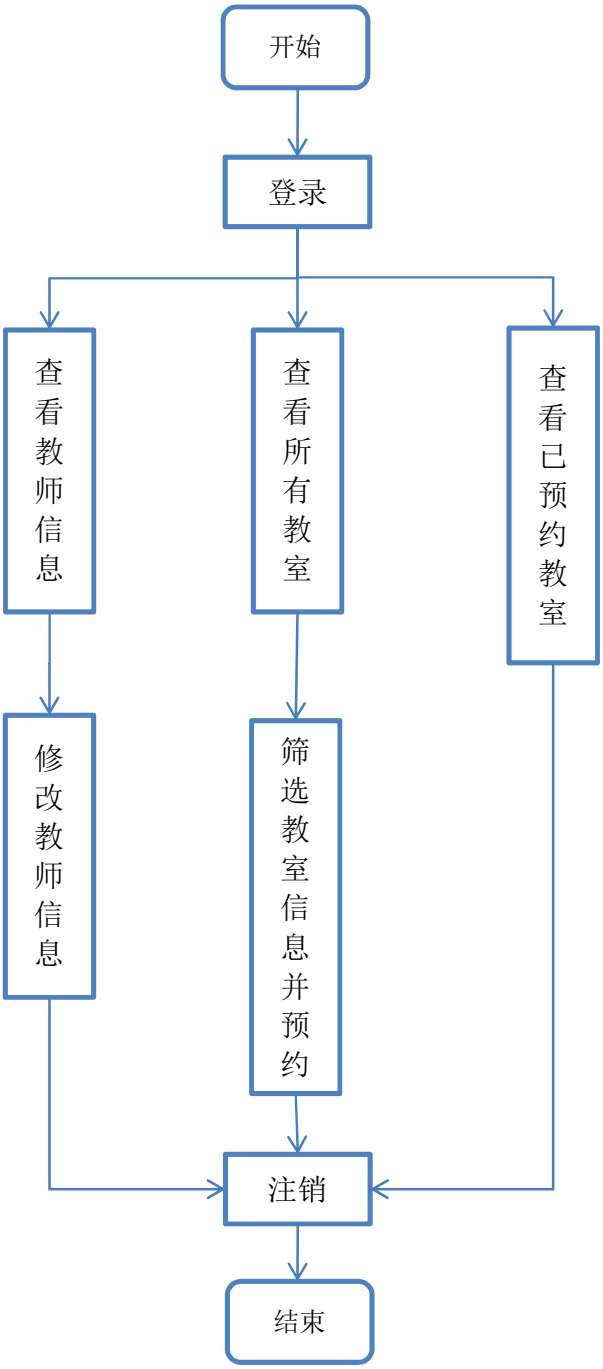

图 2.2 逻辑结构图

3 系统设计与实现

3.1 系统开发环境

硬件环境：笔记本一台

软件环境：MySQL Workbench + IntelliJ IDEA 2017

3.2 详细设计与实现

本程序采用 B/S 架构设计，使用数据、业务、视图分离的方式设计。数据层采用 MySQL 数据库，使用 MySQL Workbench 设计。包括了教室信息（teacher 表）、教室信息（classroom 表）、预约信息（hire 表）、预约时间信息（time 表）四个表。其中教师信息和教室信息表分别用来存储教室和教师的信息，预约信息表和预约时间信息表用来储存教师预约的教室信息，以及预约的时间。

4 主要源代码

完整源码请移步 github　　https://github.com/ninggc/CMS_N

数据库结构语句

```
SET FOREIGN_KEY_CHECKS=0;

-- ----------------------------
-- Table structure for classroom
-- ----------------------------
DROP TABLE IF EXISTS `classroom`;
CREATE TABLE `classroom` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `number` varchar(8) NOT NULL DEFAULT '',
  `capacity` int(11) NOT NULL DEFAULT '0',
  `extra` varchar(45) DEFAULT '',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=8 DEFAULT CHARSET=utf8;
```

```
-- ----------------------------
-- Table structure for day
-- ----------------------------
DROP TABLE IF EXISTS `day`;
CREATE TABLE `day` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `hire_id` int(11) NOT NULL,
  `day` date NOT NULL,
  PRIMARY KEY (`id`),
  KEY `fk_day_hire1_idx` (`hire_id`),
  CONSTRAINT `fk_day_hire1` FOREIGN KEY (`hire_id`) REFERENCES `hire` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8;

-- ----------------------------
-- Records of day
-- ----------------------------
INSERT INTO `day` VALUES ('2', '4', '2017-07-11');

-- ----------------------------
-- Table structure for hire
-- ----------------------------
DROP TABLE IF EXISTS `hire`;
CREATE TABLE `hire` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `teacher_id` int(11) NOT NULL,
  `classroom_id` int(11) NOT NULL,
  `time_id` int(11) DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `fk_hire_classroom1_idx` (`classroom_id`),
  KEY `fk_hire_teacher1_idx` (`teacher_id`),
  KEY `fk_hire_time1` (`time_id`),
  CONSTRAINT `fk_hire_classroom1` FOREIGN KEY (`classroom_id`) REFERENCES `classroom` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `fk_hire_teacher1` FOREIGN KEY (`teacher_id`) REFERENCES `teacher` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `fk_hire_time1` FOREIGN KEY (`time_id`) REFERENCES `time` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB AUTO_INCREMENT=31 DEFAULT CHARSET=utf8;
```

```
-- ----------------------------
-- Table structure for teacher
-- ----------------------------
DROP TABLE IF EXISTS `teacher`;
CREATE TABLE `teacher` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(8) NOT NULL DEFAULT '',
  `password` varchar(32) NOT NULL,
  `course` varchar(45) NOT NULL DEFAULT '',
  `career_title` varchar(45) NOT NULL DEFAULT '',
  PRIMARY KEY (`id`,`name`),
  UNIQUE KEY `name` (`name`)
) ENGINE=InnoDB AUTO_INCREMENT=7 DEFAULT CHARSET=utf8;

-- ----------------------------
-- Table structure for time
-- ----------------------------
DROP TABLE IF EXISTS `time`;
CREATE TABLE `time` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `day` date NOT NULL,
  `time` varchar(100) NOT NULL DEFAULT '0',
  PRIMARY KEY (`id`),
  KEY `fk_time_day1_idx` (`day`)
) ENGINE=InnoDB AUTO_INCREMENT=23 DEFAULT CHARSET=utf8;
```

```
package com.ning.DAO;

import javax.persistence.*;

/**
 * Created by ning on 2017/7/11.
 */
@Entity
@Table(name = "hire")
public class HireEntity implements IEntity {
    private int id;
    private transient TimeEntity time;
//    private Set<DayEntity> days = new HashSet<>();

    private transient ClassroomEntity classroom;
```

```
    private transient TeacherEntity teacher;

    @Id
    @Column(name = "id")
    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    @OneToOne()
    @JoinColumn(name = "time_id")
    public TimeEntity getTime() {
        return time;
    }

    public void setTime(TimeEntity timeEntity) {
        this.time = timeEntity;
    }

    @ManyToOne(cascade = CascadeType.ALL, optional = false)
//    @JoinColumn(name="hiresForClassroom",
referencedColumnName="id",insertable=false,updatable=false)
    @JoinColumn(name = "classroom_id")
    public ClassroomEntity getClassroom() {
        return classroom;
    }

    public void setClassroom(ClassroomEntity classroom) {
        this.classroom = classroom;
    }

    @ManyToOne(cascade = CascadeType.ALL, optional = false)
    @JoinColumn(name = "teacher_id")
    public TeacherEntity getTeacher() {
        return teacher;
    }

    public void setTeacher(TeacherEntity teacher) {
        this.teacher = teacher;
```

```
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;

        HireEntity that = (HireEntity) o;

        if (id != that.id) return false;

        return true;
    }

    @Override
    public int hashCode() {
        return id;
    }

    @Override
    public String toJSON() {
        return gson.toJson(this);
    }
}

package com.ning.DO;

import com.ning.DAO.ClassroomEntity;
import com.ning.DAO.HireEntity;
import com.ning.DAO.TeacherEntity;
import com.ning.DAO.TimeEntity;
import com.ning.factory.MySessionFactory;
import org.hibernate.SQLQuery;
import org.hibernate.Session;
import org.hibernate.query.Query;

import java.util.List;

/**
 * Created by ning on 2017/7/4.
 */
public class HireOperation implements IOperation<HireEntity> {
```

```
    Operation<HireEntity> o = new Operation<>();

    @Override
    public boolean delete(int id) {
        return o.delete(id, (session, integer) -> {
            HireEntity c = new HireEntity();
            c.setId(id);
            session.delete(c);
            return true;
        });
    }

    @Override
    public HireEntity selectById(int id) {
        return o.selectById(id, (session, integer) -> {
            String hql = "from HireEntity where id = :id";
            Query<HireEntity> query = session.createQuery(hql);
            query.setParameter("id", id);
            List<HireEntity> c = query.list();
            return c.get(0);
        });
    }

    public List<HireEntity> selectAll() {
        return o.selectAll((session, hireEntity) -> {
            String hql = "from HireEntity ";
            Query<HireEntity> query = session.createQuery(hql,
HireEntity.class);
            return query.list();
        });
    }

    @Deprecated
    public int selectByForeignKey(int classroom_id, int teacher_id) {
        int result = -1;
        String sql = "select * from hire where classroom_id=" + classroom_id +
" and teacher_id=" + teacher_id;
        SQLQuery query = MySessionFactory.getSession().createSQLQuery(sql);
        query.addEntity("id", HireEntity.class);
        try {
            result = ((HireEntity) query.uniqueResult()).getId();
        } catch (Exception e) {
```

```
            result = ((HireEntity) query.list().get(0)).getId();
        }
        return result;
    }

    public List<TimeEntity> selectWhichBeHired(int teacher_id, int classroom_id)
{
        return o.query(new DoInDO<String, List<TimeEntity>>() {
            @Override
            public List<TimeEntity> dosomething(Session session, String s)
throws Exception {
                String hql = "select time from HireEntity where classroom_id
= :classroom_id and teacher_id = :teacher_id";
                Query<TimeEntity> query = session.createQuery(hql,
TimeEntity.class);
                query.setParameter("teacher_id", teacher_id);
                query.setParameter("classroom_id", classroom_id);
                return query.list();
            }
        });
    }

    public boolean hireByTeacher(TeacherEntity teacher, ClassroomEntity
classroom, TimeEntity time) {
        if (teacher == null || classroom == null || time == null) {
            return false;
        }

        HireEntity hire = new HireEntity();
        hire.setClassroom(classroom);
        hire.setTeacher(teacher);
        hire.setTime(time);
        return insert(hire);
    }

    public boolean hireByTeacher(int teacher_id, int classroom_id, TimeEntity
time) {
        return o.query(new DoInDO<String, Boolean>() {
            @Override
            public Boolean dosomething(Session session, String s) throws
Exception {
                Query<TeacherEntity> queryTeacher = session.createQuery("from
```

```
TeacherEntity where id = :id", TeacherEntity.class);
                queryTeacher.setParameter("id", teacher_id);
                Query<ClassroomEntity> queryClassroom =
session.createQuery("from ClassroomEntity where id = :id",
ClassroomEntity.class);
                queryClassroom.setParameter("id", classroom_id);

                HireEntity hire = new HireEntity();
                hire.setClassroom(queryClassroom.uniqueResult());
                hire.setTeacher(queryTeacher.uniqueResult());

                String hql = "select max(id) from TimeEntity";
                Query query = session.createQuery(hql);
                if (time.getId() == 0) {
                    time.setId((Integer) query.uniqueResult() + 1);
                }
                hire.setTime(time);

                session.save(time);
                session.save(hire);
                return true;
            }
        });
    }


    public List<HireEntity> selectByClassroom(int classroom_id) {
        return o.selectListBy(String.valueOf(classroom_id), new DoInDO<String,
List<HireEntity>>() {
            @Override
            public List<HireEntity> dosomething(Session session, String s)
throws Exception {
                String hql = "from HireEntity where classroom.id = :classroom_id";
                Query<HireEntity> query = session.createQuery(hql,
HireEntity.class);
                query.setParameter("classroom_id", classroom_id);
                List<HireEntity> list = query.list();
                return list;
            }
        });
    }
    public List<HireEntity> selectByTeacher(int teacher_id) {
```

```
        return o.selectListBy(String.valueOf(teacher_id), new DoInDO<String,
List<HireEntity>>() {
            @Override
            public List<HireEntity> dosomething(Session session, String s)
throws Exception {
                String hql = "from HireEntity where teacher.id = :id";
                Query<HireEntity> query = session.createQuery(hql,
HireEntity.class);
                query.setParameter("id", teacher_id);
                List<HireEntity> list = query.list();
                return list;
            }
        });
    }
}
```

5 结论

5.1 运行结果

图 5.1 登录界面

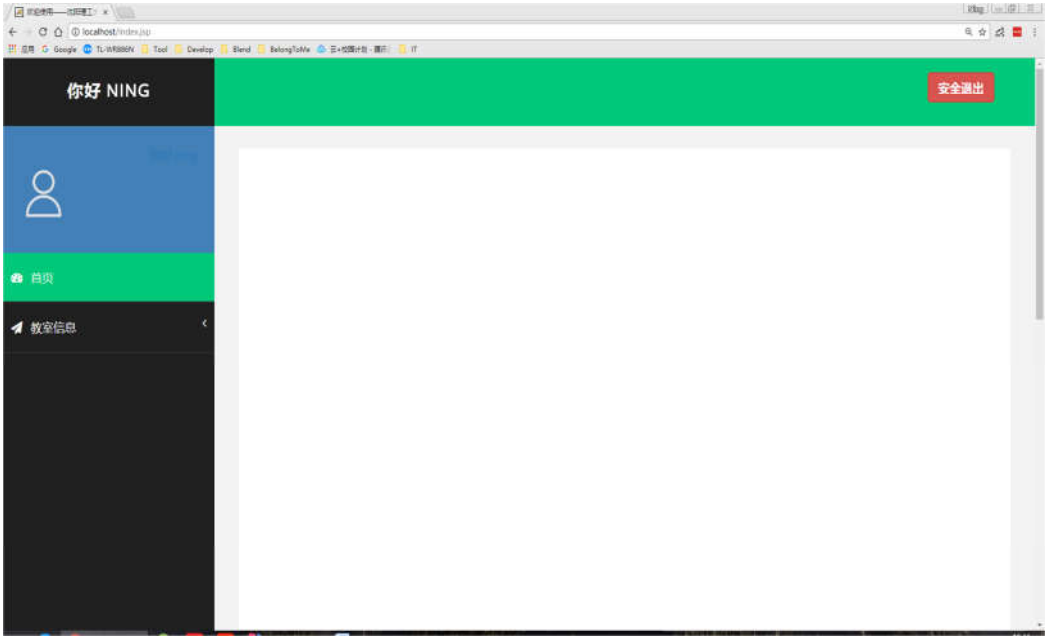

图 5.2 首页

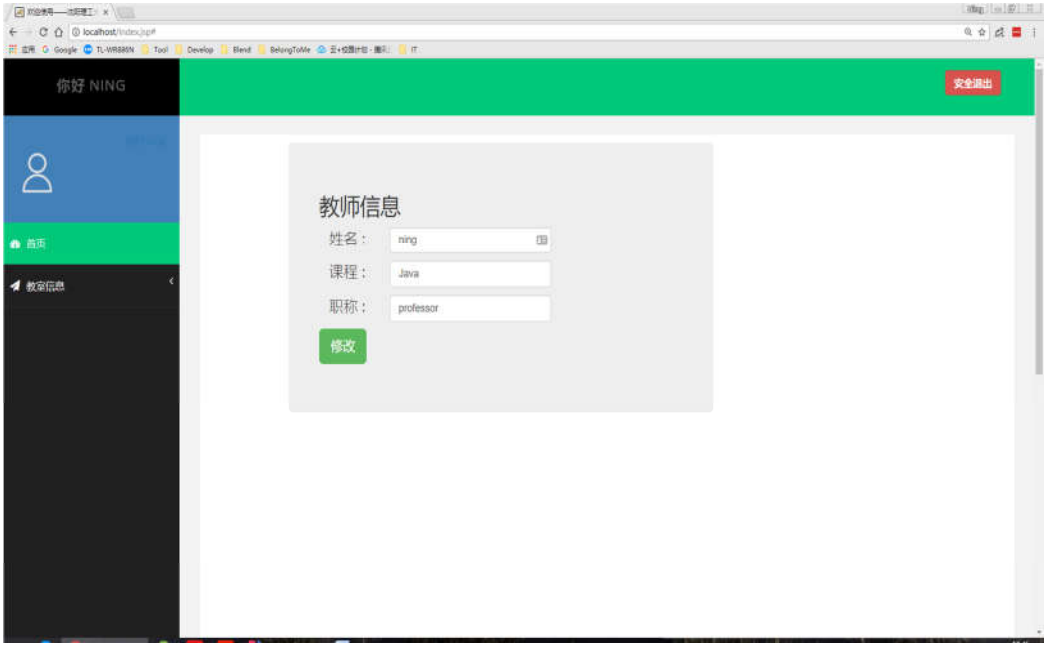

图 5.3 教师信息查看及修改界面

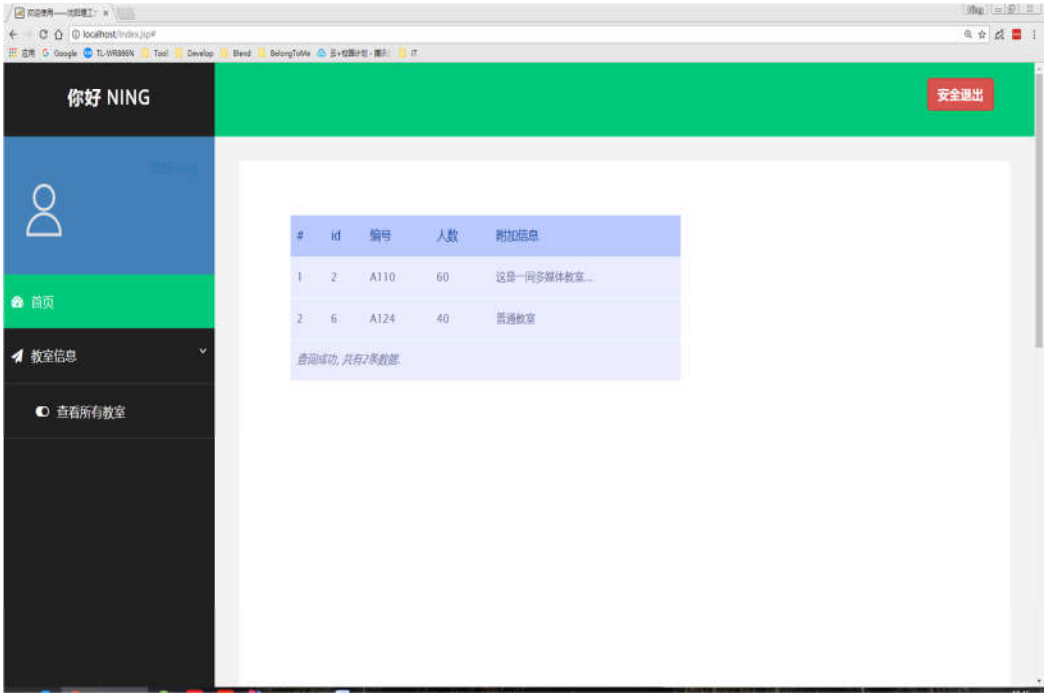

图 5.4 教室信息界面

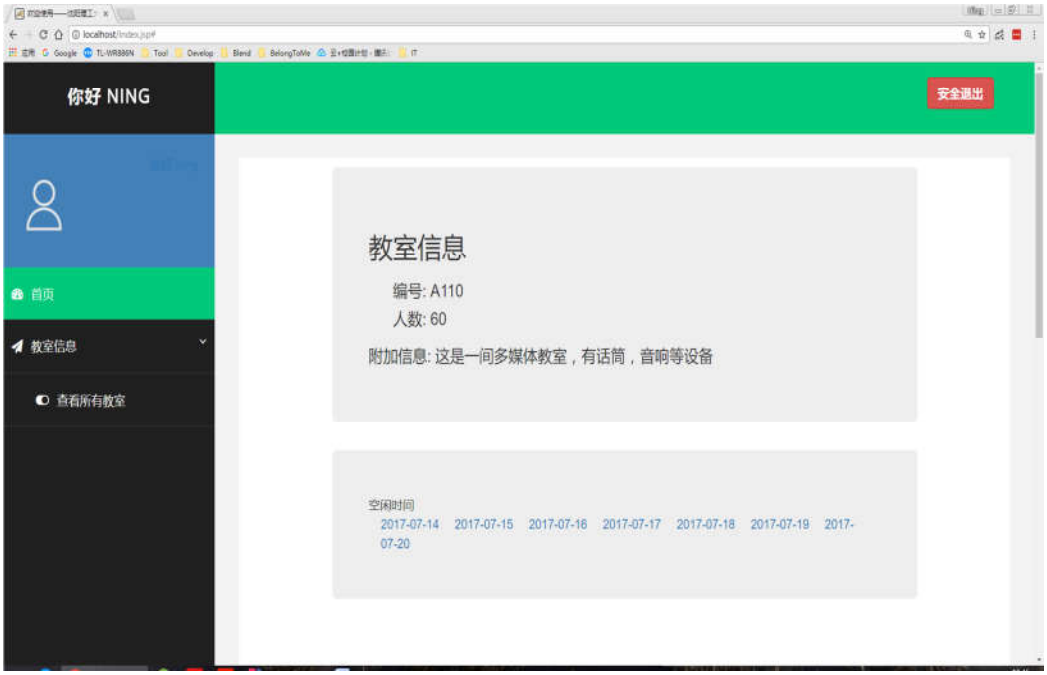

图 5.5 教室详细信息界面

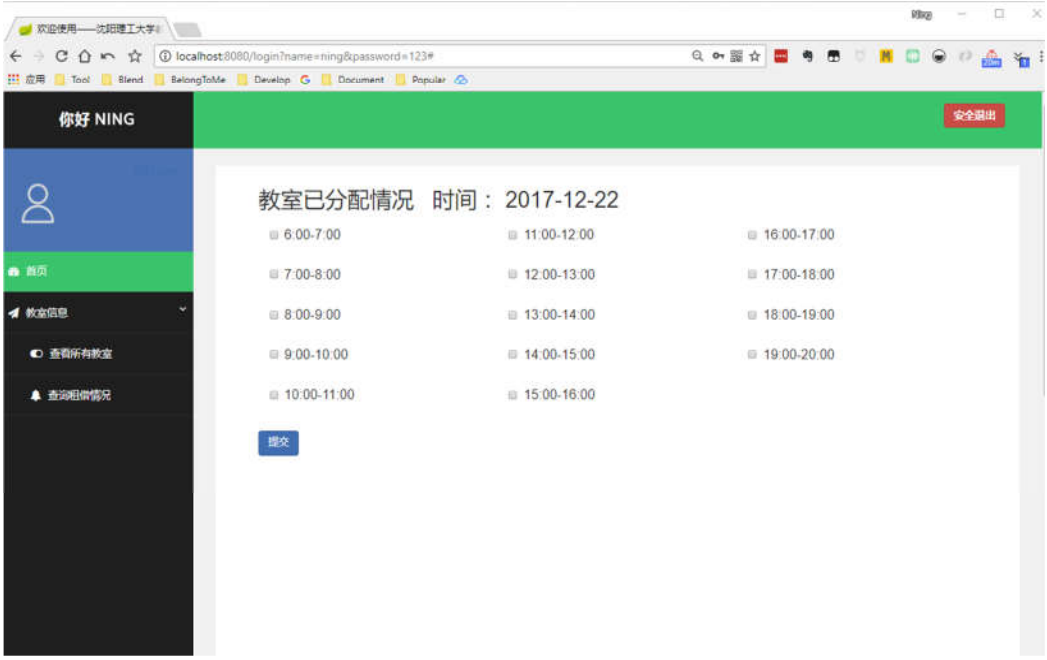

图 5.6 教室分配信息

5.2 结论

本次课程设计，让我获益匪浅，以前在学 java 和数据库时，虽然上课也好好听老师讲，但是在课后自己没有亲自动手实践过，现在有许多知识都忘记了。因此，在这次课程设计中，有许多 java 的知识我都忘记了，许多 java 函数的功能我都不会了。所以，在课程设计中我经常遇到许多问题，这次课程设计中我遇到问题，皆可以问老师或者其他会的同学。当然，许多时候我们会不停地翻 sql 与 java EE。通过这次数据库开发的课程设计，我更加深入的了解数据库这门学科，更使自己有了继续探索的兴趣。于个人而言，在程序设计的过程中，我深感“认真严谨”这个词的重要性，一点点小的马虎，便会导致整个程序不能正常运行。在今后的学习中，我定将“认真严谨”时刻作为自的谨言。与此同时，我们小组成员的互帮互助，让我体会到了团结的力量，而更让人难以忘怀的是在热烈讨论问题时，那激情横溢的场面。总之，此次课程设计在我的学生生涯中启上了至关重要的作用。

6 参考文献.

[1] 霍斯特曼，科内尔. Java 核心技术 卷一 基础知识[M]. 北京: 机械工业出版社，2013.11

[2] Ben Forta. MySQL 必知必会[M]. 北京: 人民邮电出版社，2009.1.

[3] 唐汉明. 深入浅出 MySQL[M]. 北京: 人民邮电出版社，2014.1