该系统的开发采用MVC结构，故项目整体结构如下图所示：



其中，dao层包含package com.gxun.dao.inter和package com.gxun.dao.impl，主要是实现对数据库表的增删改查操作。package com.gxun.dao.inter主要是声明接口，优化了结构，package com.gxun.dao.imp主要是实现接口。主要代码如下所示：

BaseDaoInter接口，该接口存放的是抽象出来的公有的对数据库进行操作的方法，主要是查询操作和增加操作，减少了代码的冗余度。

public interface BaseDaoInter {

/\*\*

\* 获取数据集合

\* @param type 返回对象的class

\* @param sql 要执行的sql语句

\* @return

\*/

List<Object> getList(Class type, String sql);

/\*\*

\* 获取数据集合

\* @param type 返回对象的class

\* @param sql 执行的sql语句

\* @param param Object[] 数组参数

\* @return

\*/

List<Object> getList(Class type, String sql, Object[] param);

/\*\*

\* 获取一个对象

\* @param type

\* @param sql

\* @param param

\* @return

\*/

Object getObject(Class type, String sql, Object[] param);

/\*\*

\* 插入数据

\* 开启事务

\* @param conn

\* @param sql

\* @param param

\* @throws SQLException

\*/

void insertTransaction(Connection conn, String sql, Object[] param) throws SQLException;

/\*\*

\* 插入或更新数据

\* @param conn

\* @param sql

\* @param param

\* @throws SQLException

\*/

void updateTransaction(Connection conn, String sql, Object[] param) throws SQLException;

}

BaseDaoimpl:

public class BaseDaoImpl implements BaseDaoInter{

public List<Object> getList(Class type, String sql, Object[] param) {

QueryRunner qr = new QueryRunner(MysqlTool.getDataSource());

List<Object> list = new LinkedList<>();

try {

list = qr.query(sql, new BeanListHandler(type), param);

} catch (SQLException e) {

e.printStackTrace();

}

return list;

}

public List<Object> getList(Class type, String sql) {

QueryRunner qr = new QueryRunner(MysqlTool.getDataSource());

List<Object> list = new LinkedList<>();

try {

list = qr.query(sql, new BeanListHandler(type));

String sql = "SELECT id, name customerName, email, birth " + "FROM customers WHERE id = ?";

} catch (SQLException e) {

e.printStackTrace();

}

return list;

}

public Object getObject(Class type, String sql, Object[] param) {

QueryRunner qr = new QueryRunner(MysqlTool.getDataSource());

Object obj = new LinkedList<>();

try {

obj = qr.query(sql, new BeanHandler(type), param);

} catch (SQLException e) {

e.printStackTrace();

}

return obj;

}

public void insertTransaction(Connection conn, String sql, Object[] param) throws SQLException {

try {

PreparedStatement ps = conn.prepareStatement(sql, PreparedStatement.RETURN\_GENERATED\_KEYS);

//设置值

if(param != null && param.length > 0){

for(int i = 0;i < param.length;i++){

ps.setObject(i+1, param[i]);

}

}

//执行sql语句

ps.execute();

} catch (SQLException e) {

e.printStackTrace();

}

}

public void updateTransaction(Connection conn, String sql, Object[] param) throws SQLException {

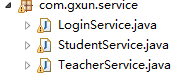
QueryRunner qr = new QueryRunner();

qr.update(conn, sql, param);

}

}

com.gxun.dao.service对应service层，该包下包含的java文件如下所示：



作用是调用相对应的dao层接口，实现页面请求的处理，并将处理结果返回给控制层。实现数据层和界面层的交互，主要示例如下所示：

/\*\*

\* 获取当前老师发布的作业

\* @param page

\* @param rows

\* @return

\*/

public String getWokeList(){

List<Object> list = dao.getList(WorkBean.class,

"SELECT \* FROM work");

String result = JSONArray.fromObject(list).toString();

return result;

}

/\*\*

\* 作业提交

\* @param page

\* @param rows

\* @return

\*/

public void addSubmit(SubmitBean submit) {

try {

//开启事务

MysqlTool.startTransaction();

String sql = "INSERT INTO submit( wid, tid, sid, answer,qid) value(?,?,?,?,?)";

Object[] param = new Object[]{

submit.getWid(),

submit.getTid(),

submit.getSid(),

submit.getAnswer(),

submit.getQid()

};

//添加教师信息

submitDao.addSubmit(sql, param);

//提交事务

MysqlTool.commit();

} catch (Exception e) {

//回滚事务

MysqlTool.rollback();

e.printStackTrace();

} finally {

MysqlTool.closeConnection();

}

}

/\*\*

\* 作业发布

\* @param page

\* @param rows

\* @return

\*/

public void addWork (WorkBean work,String[] ids) {

Connection conn = MysqlTool.getConnection();

try {

//开启事务

MysqlTool.startTransaction();

String sql = "INSERT INTO work( wname, starttime, endtime, tid) value(?,?,?,?)";

Object[] param = new Object[]{

work.getWname(),

work.getStarttime(),

work.getEndtime(),

work.getTid()

};

//添加作业信息

dao.insertTransaction(conn, sql, param );

MysqlTool.commit();

//添加作业与题目关系

WorkBean work1 = new WorkBean();

work1 = (WorkBean)dao.getObject(WorkBean.class,"SELECT \* FROM work WHERE wname=?", new Object[]{work.getWname()});

for(int i=0;i<ids.length;i++)

dao.insertTransaction(conn, "INSERT INTO wq(wid,qid) value(?,?)", new Object[]{

work1.getWid(),

ids[i]

});

initMark(work1.getWid());

//提交事务

MysqlTool.commit();

} catch (Exception e) {

//回滚事务

MysqlTool.rollback();

e.printStackTrace();

}

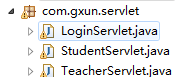
finally {

MysqlTool.closeConnection();

}

}

package com.gxun.dao.servlet对应控制层，该包下的java文件如下所示：



主要是将页面的请求传给service进行处理，并将service层的处理结果传给界面，是界面和逻辑层的接口。主要代码示例如下：

/\*\*

\* 教师类TeacherServlet

\*

\*/

public class TeacherServlet extends HttpServlet {

//创建服务层对象

private TeacherService service = new TeacherService();

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

//获取请求的方法

String method = request.getParameter("method");

if("LoginOut".equals(method)){ //退出系统

loginOut(request, response);

}else if("toTeacherPubView".equalsIgnoreCase(method)){ //转发到作业发布

request.getRequestDispatcher("/WEB-INF/view/teacher/TeacherPub.jsp").forward(request, response);

} else if("toTeacherWorkCorrectView".equalsIgnoreCase(method)){ //转发到作业批改

request.getRequestDispatcher("/WEB-INF/view/teacher/TeacherWorkCorrect.jsp").forward(request, response);

} else if("toTeacherPersonalView".equalsIgnoreCase(method)){ //转发到个人信息

request.getRequestDispatcher("/WEB-INF/view/teacher/TeacherPersonal.jsp").forward(request, response);

}else if("toQueryEvenView".equalsIgnoreCase(method)){ //转发到查询成绩列表页

request.getRequestDispatcher("/WEB-INF/view/teacher/TeacherQuery.jsp").forward(request, response);

}

}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

//获取请求的方法

String method = request.getParameter("method");

if("getQuestionList".equalsIgnoreCase(method)){ //获取所有教师数据

getQuestionList(request, response);

} else if("addWork".equalsIgnoreCase(method)){ //添加教师

addWork(request, response);

}else if("deleteQuestion".equalsIgnoreCase(method)) {

deleteQuestion(request, response);

}else if("correctSub".equalsIgnoreCase(method)) {

correctSub(request, response);

}else if("addQuestion".equalsIgnoreCase(method)) {

addQuestion(request, response);

}else if("getGradeList".equalsIgnoreCase(method)) {

getGradeList(request, response);

}else if("getWorkList".equalsIgnoreCase(method)) {

getWorkList(request, response);

}

else if("getSubmitList".equalsIgnoreCase(method)) {

getSubmitList(request, response);

}

}

private void getQuestionList(HttpServletRequest request, HttpServletResponse response) throws IOException {

//获取数据

String result = service.getQuestionList();

//返回数据

response.getWriter().write(result);

}

private void addWork(HttpServletRequest request, HttpServletResponse response) throws IOException {

//作业发布

WorkBean work = new WorkBean();

work.setTid(((TeacherBean)request.getSession().getAttribute("account")).getTid());

work.setWname(request.getParameter("wname"));

work.setStarttime(request.getParameter("starttime"));

work.setEndtime(request.getParameter("endtime"));

//获取作业的题号

String[] ids = request.getParameterValues("ids[]");

try {

service.addWork(work,ids);

response.getWriter().write("success");

} catch (Exception e) {

response.getWriter().write("fail");

e.printStackTrace();

}

}

private void correctSub(HttpServletRequest request, HttpServletResponse response)throws IOException {

int grade = Integer.parseInt(request.getParameter("grade"));

String[] wids = request.getParameterValues("wids[]");

String[] qids = request.getParameterValues("qids[]");

String[] sids = request.getParameterValues("sids[]");

int wid = Integer.parseInt(wids[0]);

int qid = Integer.parseInt(qids[0]);

String sid = sids[0];

try {

service.updateSubmit(grade,wid,qid,sid);

response.getWriter().write("success");

} catch (Exception e) {

response.getWriter().write("fail");

e.printStackTrace();

}

}

private void addQuestion(HttpServletRequest request, HttpServletResponse response) throws IOException {

//增加题目

QuestionBean question = new QuestionBean();

question.setQcontent(request.getParameter("qcontent"));

question.setSnum(Integer.parseInt(request.getParameter("snum")));

try {

service.addQuestion(question);

response.getWriter().write("success");

} catch (Exception e) {

response.getWriter().write("fail");

e.printStackTrace();

}

}

private void deleteQuestion(HttpServletRequest request, HttpServletResponse response)throws IOException {

//获取要删除的题号

String[] qids = request.getParameterValues("qids[]");

try {

service.deleteQuestion(qids);

response.getWriter().write("success");

} catch (Exception e) {

response.getWriter().write("fail");

e.printStackTrace();

}

}

private void getGradeList(HttpServletRequest request, HttpServletResponse response)throws IOException {

//获取作业号

String wid=request.getParameter("wid");

try {

service.statistical();

//获取数据

String result=service.getMarkList(wid);

response.getWriter().write(result);

} catch (Exception e) {

response.getWriter().write("fail");

e.printStackTrace();

}

}

private void getWorkList(HttpServletRequest request, HttpServletResponse response)throws IOException {

//获取老师工号

TeacherBean teacher=(TeacherBean) request.getSession().getAttribute("account");

try {

//获取数据

String result=service.getWokeList(teacher.getTid());

response.getWriter().write(result);

} catch (Exception e) {

response.getWriter().write("fail");

e.printStackTrace();

}

}

private void getSubmitList(HttpServletRequest request, HttpServletResponse response)throws IOException {

//获取作业号

String wid=request.getParameter("wid");

System.out.println(wid);

try {

//获取数据

String result=service.getSubmitList(wid);

response.getWriter().write(result);

} catch (Exception e) {

response.getWriter().write("fail");

e.printStackTrace();

}

}

/\*\*

\* 退出系统

\* @param request

\* @param response

\* @throws IOException

\* @throws ServletException

\*/

private void loginOut(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

//退出系统时清除系统登录的用户

request.getSession().removeAttribute("user");

String contextPath = request.getContextPath();

//转发到登录界面

response.sendRedirect(contextPath+"/index.jsp");

}

}

界面实现主要采用easyui和h-ui框架，代码示例略。

