

# 停车场管理系统实验报告

---

## 目 录

<b>1.绪论</b>	<b>1</b>
1.1 背景	2
1.2 知识框架	3
<b>2.需求分析</b>	<b>4</b>
<b>3.概要设计</b>	<b>4</b>
3.1 系统功能设计	2
3.2 系统业务流程	2
3.3 数据库设计	2
3.4 项目中的 MVC 结构	2
<b>4.详细设计</b>	<b>4</b>
4.1 创建 DBUtil 层	2
4.2 创建 Service 层	2
4.2.1 IC 卡业务操作	6
4.2.2 角色的业务操作	6
4.2.3 固定车位的逻辑业务	6
4.2.4 临时车位逻辑业务	6
4.2.5 车位的逻辑业务	6
4.3 创建 Servlet 层	2
4.3.1 固定车位的数据处理	6
4.3.2 临时车位数据处理	6
4.4 页面设计	2
4.4.1 登录页面设计	6
4.4.2 主页面设计	6
4.4.3 滑动效果设计	6
<b>5.心得体会</b>	<b>4</b>
<b>6.参考文献</b>	<b>4</b>

## 一、 绪论

### 1.1 背景

智能化停车场管理系统是针对地域小区车辆管理开发的一个系统，随着世界人口的增多，车辆的增多，车辆管理系统在现在生活中成为了重要的一部分，世界上各个小区，社区需要便于管理的系统去管理各个地方的车辆信息，来提高人们的生活工作效率，同时减少不必要的麻烦，本车辆管理系统满足了社会的普遍需求。

### 1.2 知识框架

JDBC 连接数据库、JavaBean 封装业务逻辑、Jsp、Css、JS、Servlet、HTML、三层构造开发模式。

## 二、 需求分析

IT 行业的高速发展让计算机技术深入日常生活的每一个细节，在各个领域中，计算机技术的应用帮助人们减少劳动量，提高工作效率，发挥着越来越重要的作用。随着城市化程度的加深，房地产行业日益兴盛，越来越多的停车场散布在城市里，停车场中来往的车辆与日俱增，对停车场的管理也是非常重要。本讨论组结合此次的课程设计开发以下的停车场管理系统，使停车场里的车辆能得到有序并且相对全面的管理。

需求：主要是对小区的车位、车主信息、IC 卡等以信息的方式进行管理，极大地减少停车在各个环节可能出现地失误和意外。使管理更加方便。

根据需求描述开发细节如下图：



图 2.1.1 停车场管理系统开发细节图

经过分析描绘出预备知识导图：

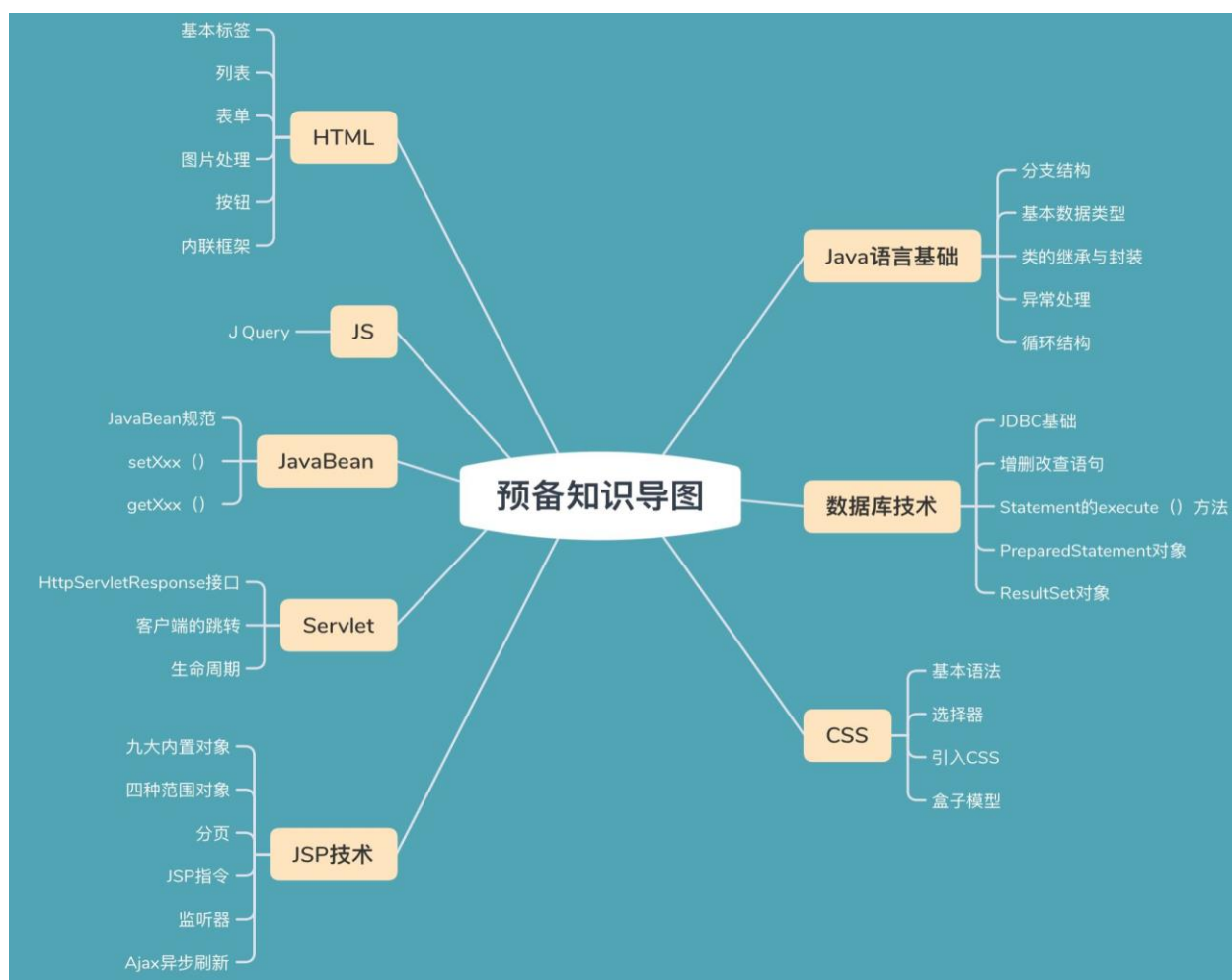


图 2.1.2 预备知识导图

## 三、概要设计（总体设计）

### 3.1 系统功能设计

智能化停车场管理系统功能结构图如下图：

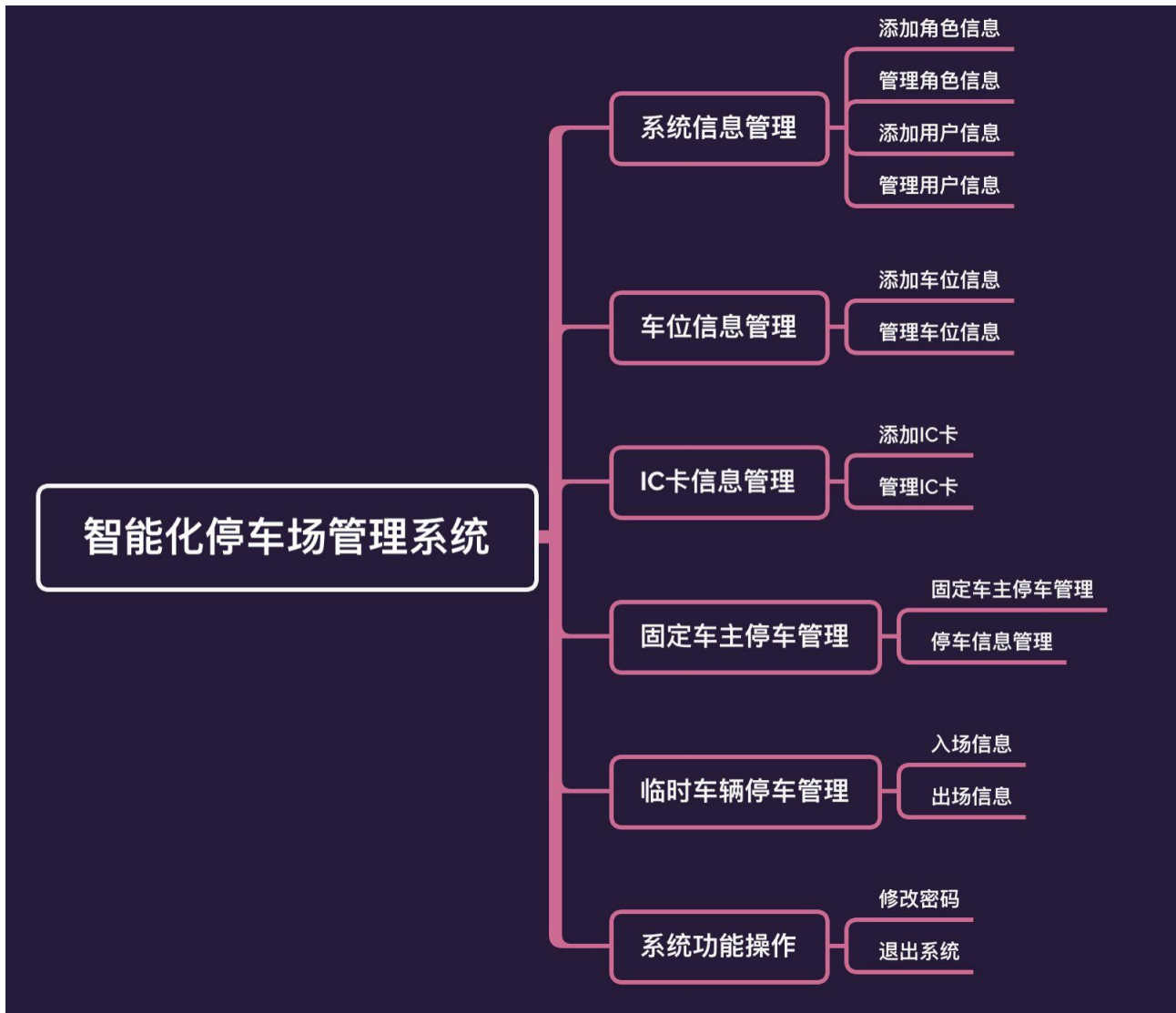
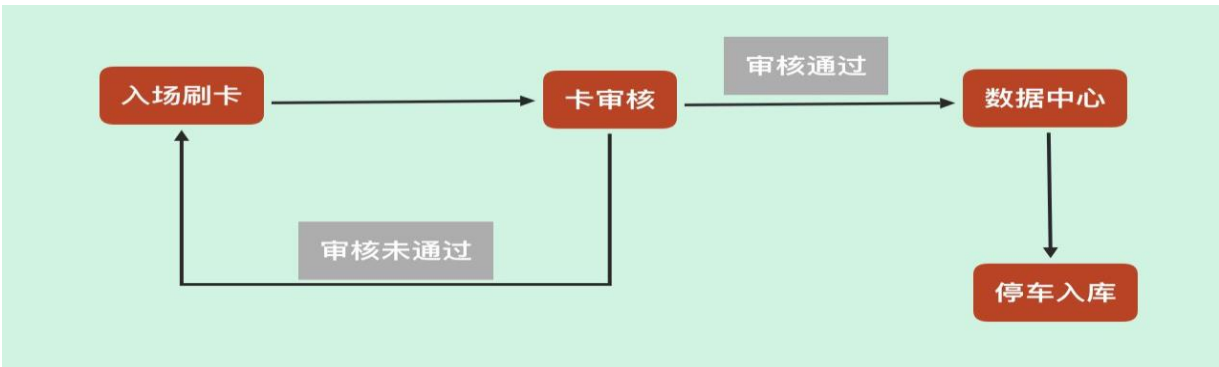


图 3.1.1 停车场管理系统功能结构图

### 3.2 系统业务流程

停车场管理系统业务流程图如下：



3.3 数据库设计

智能化停车场管理系统采用 MySQL 作为后台数据库，数据库名为 parking，详细信息如下图：

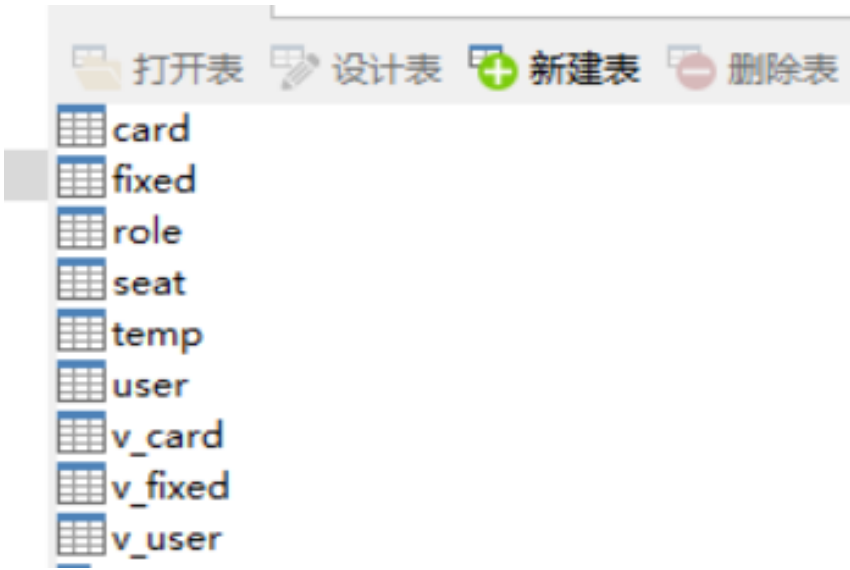
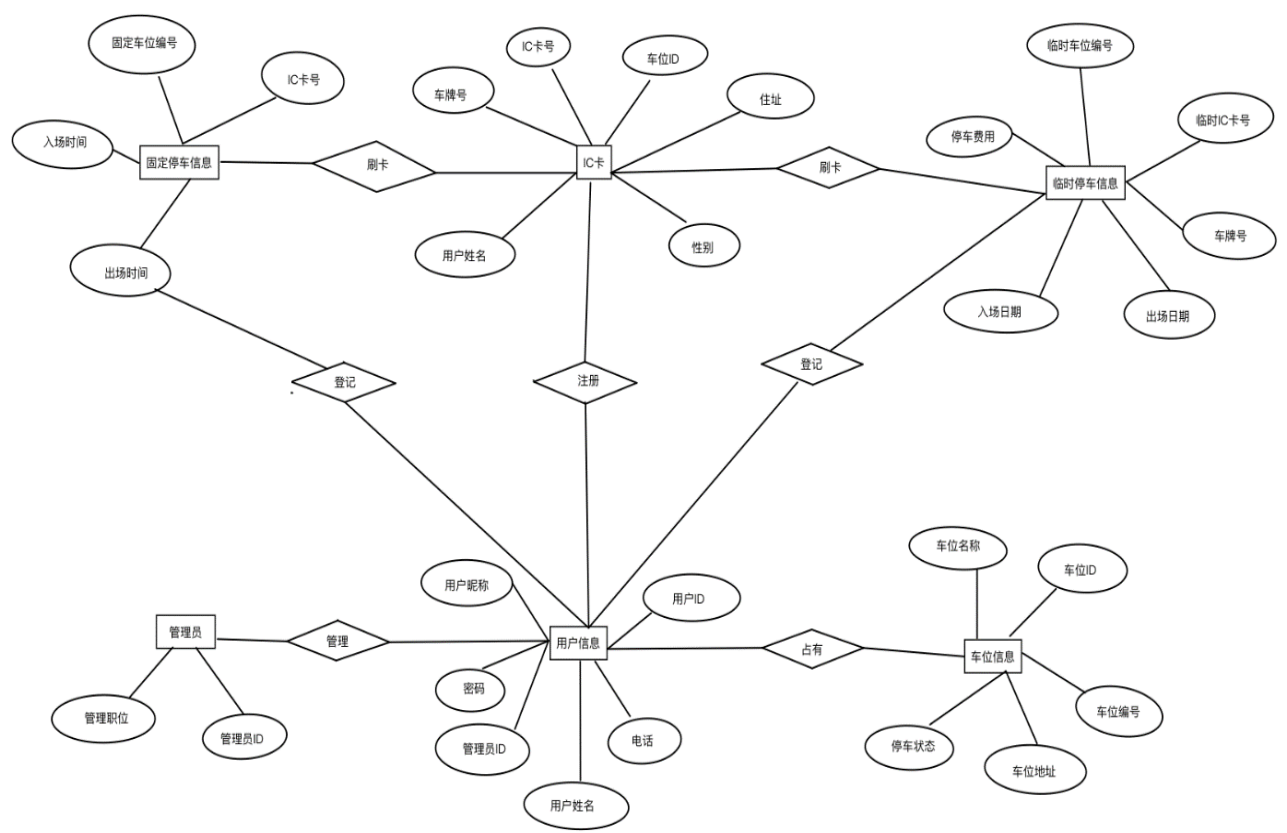


图 3.1.1 数据库结构图

数据库需求分析图：

# 停车场管理系统实验报告




## 3.3.1 设计数据表

①IC 卡信息表（IC 卡号, 车位 ID, 车牌号, 住址, 性别, 用户姓名）


栏位	索引	外键	触发器	选项	注释	SQL 预览
名					类型	长度
card_id					varchar	50
seat_id					varchar	50
user_name					varchar	50
user_gender					varchar	1
user_addr					varchar	50
car_num					varchar	50

②用户信息：（用户昵称, 密码, 管理员 ID, 用户姓名, 电话, 用户 ID）


## 停车场管理系统实验报告

名	类型	长度	小数点	不是 null	
▶ user_id	varchar	50	0	<input checked="" type="checkbox"/>	 1
role_id	varchar	50	0	<input checked="" type="checkbox"/>	
user_name	varchar	50	0	<input checked="" type="checkbox"/>	
real_name	varchar	50	0	<input checked="" type="checkbox"/>	
user_pwd	varchar	20	0	<input checked="" type="checkbox"/>	
user_phone	varchar	50	0	<input type="checkbox"/>	

③固定停车信息：（固定车位编号，IC 卡号，入场日期，出场日期）

名	类型	长度	小数点	不是 null	
fixed_id	varchar	50	0	<input checked="" type="checkbox"/>	 1
▶ card_id	varchar	50	0	<input checked="" type="checkbox"/>	
entry_date	date	0	0	<input checked="" type="checkbox"/>	
entry_time	time	0	0	<input checked="" type="checkbox"/>	
out_date	date	0	0	<input type="checkbox"/>	
out_time	time	0	0	<input type="checkbox"/>	

④临时停车信息：（停车费用，车位编号，临时 IC 卡号，车牌号，入场日期，出场日期）

名	类型	长度	小数点	不是 null	
temp_id	varchar	50	0	<input checked="" type="checkbox"/>	 1
▶ card_id	varchar	50	0	<input checked="" type="checkbox"/>	
car_num	varchar	50	0	<input checked="" type="checkbox"/>	
entry_date	date	0	0	<input checked="" type="checkbox"/>	
entry_time	time	0	0	<input checked="" type="checkbox"/>	
out_date	date	0	0	<input type="checkbox"/>	
out_time	time	0	0	<input type="checkbox"/>	
temp_money	float	0	0	<input type="checkbox"/>	

⑤车位信息：（车位 ID,车位编号，车位地址，停车状态，车位名称）

名	类型	长度	小数点	不是 null	
▶ seat_id	varchar	50	0	<input checked="" type="checkbox"/>	 1
seat_num	varchar	50	0	<input checked="" type="checkbox"/>	
seat_section	varchar	50	0	<input checked="" type="checkbox"/>	
seat_state	int	11	0	<input checked="" type="checkbox"/>	
seat_tag	varchar	50	0	<input type="checkbox"/>	

⑥管理员：（管理职位，管理员 ID）

## 停车场管理系统实验报告

名	类型	长度	小数点	不是 null	
▶ role_id	varchar	50	0	<input checked="" type="checkbox"/>	🔑 1
role_name	varchar	50	0	<input checked="" type="checkbox"/>	

### 3.4 项目中的 MVC 结构

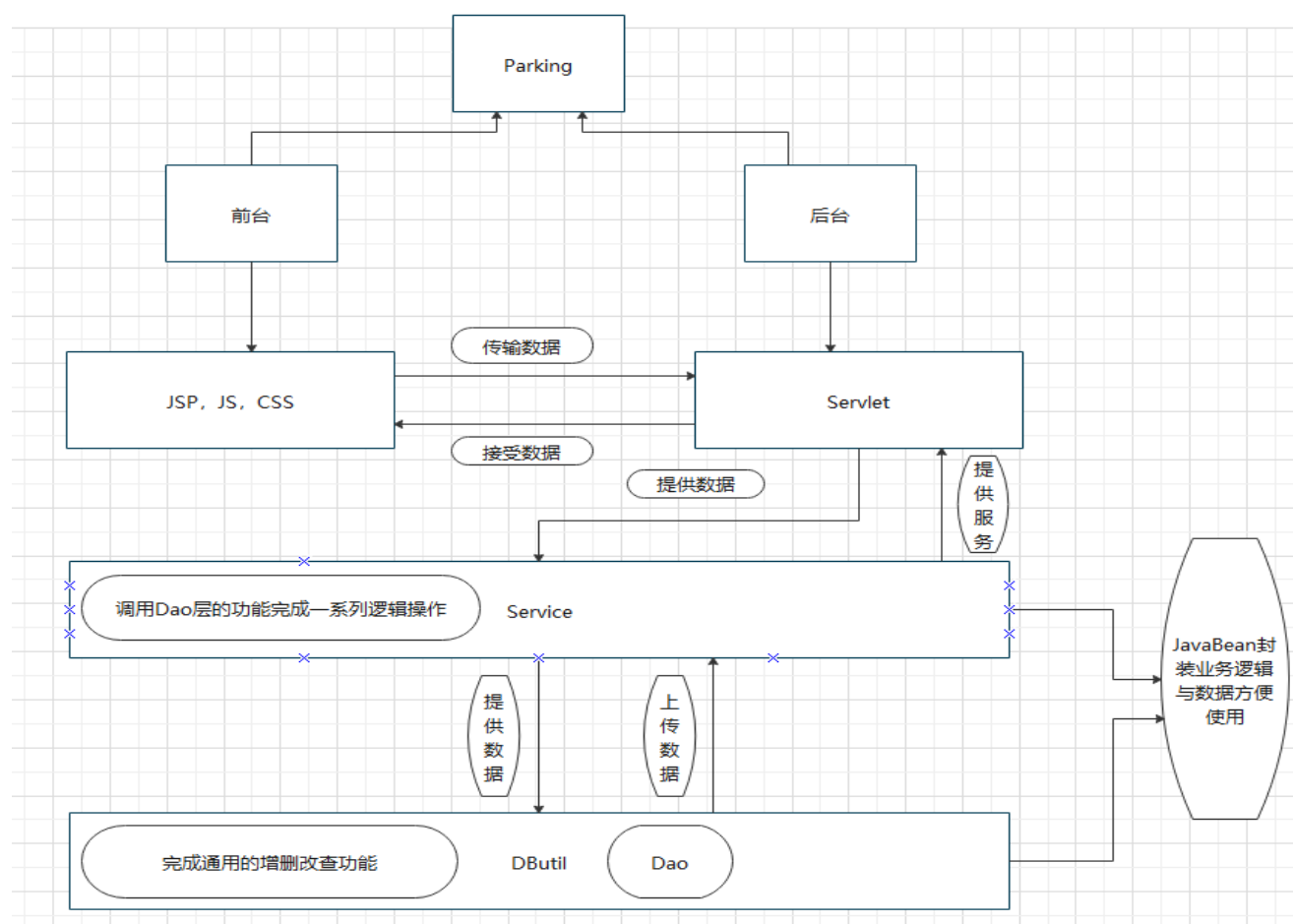


图 3.4.1 三层构造模型图

其中 MVC, M 层对应的是 DBUtil 和 Service 层, V 层对应的是前台, C 层对应的是 Servlet。  
对应 MVC 图如下所示:



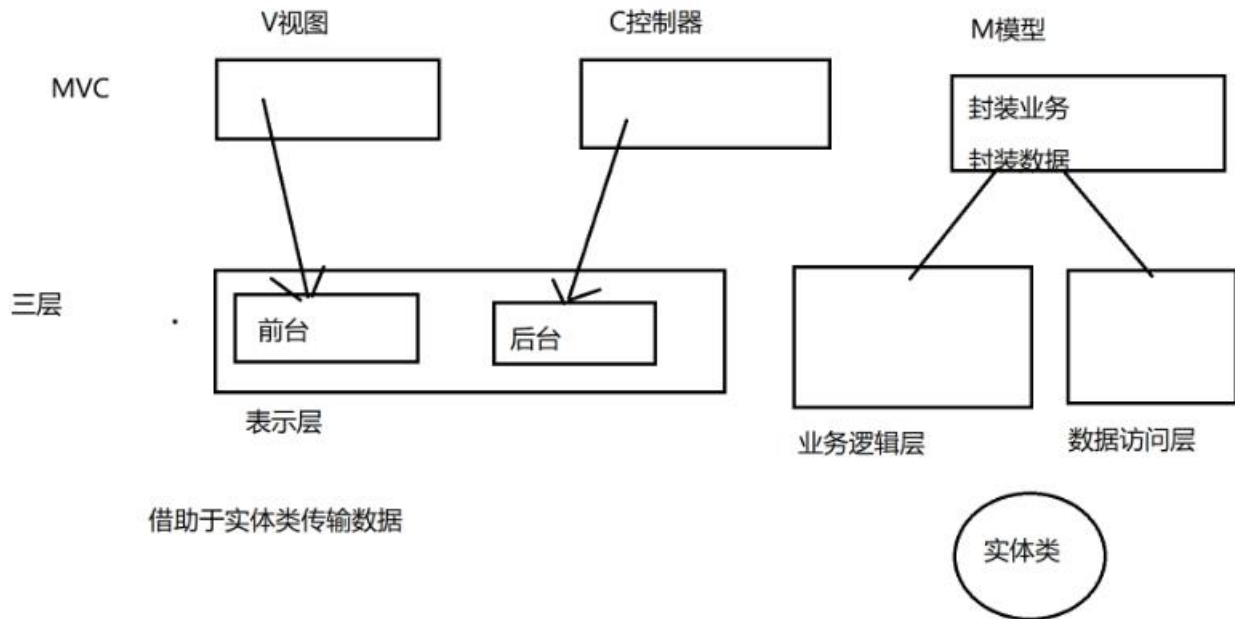


图 3.4.2 三层构造对应的 MVC 图

## 四、详细设计

### 4.1 创建 DBUtil 层

在停车场管理系统中，DBUtil 层完成访问数据库，用来实现数据库的驱动、连接和以及操作数据表。其关键代码如下：

因为每次对数据库操作之前需要连接数据库，所以讲连接数据库的部分进行提取：

```
public class ConnectionFactory {
    private static String user;
    private static String pwd;
    private static String url;
    private static String driver;
    static{
        InputStream
        iStream=ConnectionFactory.class.getClassLoader().getResourceAsStream("DBU
        til/db.properties");//读取db.properties配置文件
        Properties prop=new Properties();
        try {
            prop.load(iStream);
            user=prop.getProperty("user");//获取配置文件中的user对应的值
            pwd=prop.getProperty("pwd");//获取配置文件中的pwd对应的值
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

## 停车场管理系统实验报告

```
url=prop.getProperty("url");//获取配置文件中的url对应的值
driver=prop.getProperty("driver");//获取配置文件中的driver对应的
```

值

```
    } catch (Exception e) {
        e.printStackTrace();
    }
}
public static Connection getConnection()
{
    Connection conn=null;
    try {
        Class.forName(driver);//数据库驱动注册
        conn=DriverManager.getConnection(url,user,pwd);//获取数据库链接
    } catch (Exception e) {
        e.printStackTrace();
    }
    return conn;
}
}
```

对象

对数据库的操作:

```
public class SQLUtil {
    @SuppressWarnings("finally")
    //执行非查询语句(delete、update、insert)
    public static int executeNonQuery(String sqlCmd,Object[] objList)
    {
```

```
        Connection conn=null;//数据库连接对象
        Statement sm=null;//可执行sql语句的Statement对象
        int result=-1;//返回结果, -1为数据操作失败, 非-1为操作成功
        try {
```

```
            conn=DBUtil.ConnectionFactory.getConnection();//获取数据库连接对象
            if(objList!=null)//判断是否有传入参数(也就是判断 Object[] objList
            是否为空)
            {
                PreparedStatement
```

象

是否为空)

## 停车场管理系统实验报告

---

```
pStatement=conn.prepareStatement(sqlCmd);//创建可执行带参数SQL命令
pStatement对象
    //对参数进行赋值
    for(int i=0;i<objList.length;i++)
    {
        pStatement.setObject(i+1, objList[i]);//获取传入参数的值
    }
    result=pStatement.executeUpdate();//执行相应命令
}
else { //采用字符串拼接方式
    conn=DBUtil.ConnectionFactory.getConnection();//获取数据库连接对象

    sm=conn.createStatement();//创建可执行sql语句的Statement对象
    result=sm.executeUpdate(sqlCmd);//执行相应sql命令
}
} catch (Exception e) {
    e.printStackTrace();
}
finally{
    DBUtil.CloseFactory.close(conn,sm);//关闭相应资源
    return result;
}
}
```

```
@SuppressWarnings("finally")
//执行查询操作 返回List型的数据集合 （如select *from table/select
name,age from table）
public static List<Object> executeQuery(String sqlCmd,Object[]
objList)
{
    Connection conn=null;
    Statement sm=null;
    ResultSet rSet=null;
    List<Object> list=new ArrayList<Object>();
    try
    {
        conn=DBUtil.ConnectionFactory.getConnection();//获取数据库连接对
```

## 停车场管理系统实验报告

象

```
    if(objList!=null)//判断是否有参数传入
    {
        PreparedStatement
pStatement=conn.prepareStatement(sqlCmd);//生成PreparedStatement, 用于执行
T-SQL命令
```

```
        for(int i=0;i<objList.length;i++)
        {
            pStatement.setObject(i+1, objList[i]);
        }
        rSet=pStatement.executeQuery();//执行查询命令, 返回ResultSet
        ResultSetMetaData rsmdData=rSet.getMetaData();
        int column=rsmdData.getColumnCount();//获取返回的单条数据的字
```

段数目

```
        while(rSet.next())
        {
            Object[] object=new Object[column];//对象数值, 用于作为获取
```

单条数据的载体

```
            for(int i=1;i<=column;i++)
            {
                object[i-1]=rSet.getObject(i);
            }
            list.add(object);//将获取的数据添加到集合中
        }
    }
}
```

```
else {
    System.out.println("我进入空的之中了");
    //conn=DBUtil.ConnectionFactory.getConnection();//获取数据库
```

连接对象

```
    sm=conn.createStatement();//创建Statement对象
    rSet=sm.executeQuery(sqlCmd);//执行查询查询命令
    System.out.println("rSet =" +rSet);
    ResultSetMetaData rsmdData=rSet.getMetaData();
    int column=rsmdData.getColumnCount();//获取单条数据中属性个数
    (如 select name,age from employee) 属性为: name,age
    while(rSet.next())
    {
        Object[] object=new Object[column];
```

## 停车场管理系统实验报告

---

```
        for(int i=1;i<=column;i++)
        {
            object[i-1]=rSet.getObject(i);
        }
        list.add(object);//将相应数据添加到集合中
    }
}
}
catch(Exception e)
{
    e.printStackTrace();
    list=null;
}
finally{
    DBUtil.CloseFactory.close(conn,sm,rSet);
    return list;
}
}
```

//执行标量操作，返回首行首列的数据     `select count(*) from table/select name from table`

```
@SuppressWarnings("finally")
public static Object excuteScalar(String sqlCommand,Object[] objList)
{
    Connection conn=null;
    Statement sm=null;
    ResultSet rSet=null;
    Object obj=null;
    try {
        conn=DBUtil.ConnectionFactory.getConnection();
        if(objList!=null)
        {
            PreparedStatement pStatement=conn.prepareStatement(sqlCmd);
            for(int i=0;i<objList.length;i++)
            {
                pStatement.setObject(i+1, objList[i]);
            }
        }
    }
```

```
    }
    rSet=pStatement.executeQuery();
    while(rSet.next())
    {
        obj=rSet.getObject(1);
        break;
    }
}
else {
    conn=DBUtil.ConnectionFactory.getConnection();
    sm=conn.createStatement();
    rSet=sm.executeQuery(sqlCmd);
    while(rSet.next())
    {
        obj=rSet.getObject(1);
        break;
    }
}
} catch (Exception e) {
    e.printStackTrace();

}
finally{
    CloseFactory.close(conn,sm,rSet);
    return obj;
}
}
```

操作结束后需要统一对关闭类对象：

```
public class CloseFactory {

    //关闭Connection conn,Statement sm
    public static void close(Connection conn,Statement sm)
    {
        close(conn);
        close(sm);
    }
}
```

```
//关闭Connection conn,Statement sm,ResultSet rs
public static void close(Connection conn,Statement sm,ResultSet rs)
{
    close(conn);
    close(sm);
    close(rs);
}
```

```
//关闭数据库链接对象
public static void close(Connection conn)
{
    try {
        if(conn!=null)
        {
            conn.close();
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

```
//关闭Statement对象
public static void close(Statement sm)
{
    try {
        if(sm!=null)
        {
            sm.close();
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

```
//关闭ResultSet对象
public static void close(ResultSet rs)
```

```
{
    try {
        if(rs!=null)
        {
            rs.close();
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}

}
```

此三部分分开写，极大的减少了代码冗杂，提高了开发效率，看似简单一步，却十分重要。

## 4.2 创建 Service 层

Service 层对 DBUtil 层中对数据原子性操作进行了一个组装，完成对数据一系列的逻辑业务功能的实现，减少了代码量。

### 4.2.1 IC 卡业务操作

```
public class Card {

    //获取IC卡表信息列表
    public List<Object> getEntity()
    {
        String sqlCmd="select *from Card";
        return DBUtil.SQLUtil.executeQuery(sqlCmd, null);//执行查询操作
    }

    //获取分页后IC卡表信息列表
    public List<Object> getEntity(int page)
    {
        int size=(page-1)*15;
        String sqlCmd="select *from V_Card limit "+size+",15";
        return DBUtil.SQLUtil.executeQuery(sqlCmd, null);//执行查询操作
    }
}
```



## 停车场管理系统实验报告

---

```
}

    //根据查询条件sqlWhere获取分页后IC卡表信息列表
    public List<Object> getEntityByWhere(String sqlWhere,int page)
    {
        int size=(page-1)*15;
        String sqlCmd="select *from V_Card where "+sqlWhere+" limit "+
size+",15";
        return DBUtil.SQLUtil.executeQuery(sqlCmd, null);//执行查询操作
executeQuery
    }

    //删除IC卡表信息
    public int deleteEntity(String card_id)
    {
        String sqlCmd="delete from Card where card_id='"+card_id+"'";
        return DBUtil.SQLUtil.executeNonQuery(sqlCmd, null);//执行非查
询操作executeNonQuery
    }

    //根据IC卡表编号获取IC卡表信息
    public List<Object> getEntityById(String card_id)
    {
        String sqlCmd="select *From V_Card where
card_id='"+card_id+"'";
        return DBUtil.SQLUtil.executeQuery(sqlCmd, null);//执行查询操作
executeQuery
    }

    //更新IC卡表信息
    public int updateEntity(String card_id,String seat_id,String
user_name,String user_gender,String user_addr,String car_num)
    {
        String sqlCmd="Update Card set seat_id='" + seat_id +
"',user_name='" + user_name + "',user_gender='" + user_gender +
"',user_addr='" + user_addr + "',car_num='" + car_num + "' where
card_id='"+card_id+"'";
        return SQLUtil.executeNonQuery(sqlCmd, null);
    }
}
```

```
}
```

```
//插入IC卡表信息
```

```
public int insertEntity(String card_id,String seat_id,String  
user_name,String user_gender,String user_addr,String car_num)
```

```
{
```

```
    String sqlCmd="Insert into Card values('" + card_id + "','" +  
seat_id + "','" + user_name + "','" + user_gender + "','" + user_addr +  
"','"+car_num+"')";
```

```
    return SQLUtil.executeNonQuery(sqlCmd, null);
```

```
}
```

```
//检查插入主键是否重复
```

```
public boolean checkExist(String card_id)
```

```
{
```

```
    String sqlCmd="select count(*) from V_Card where  
card_id='"+card_id+"'";
```

```
    if(1==Integer.parseInt(SQLUtil.excuteScalar(sqlCmd,  
null).toString())) )
```

```
    {
```

```
        return true;
```

```
    }
```

```
    return false;
```

```
}
```

```
//获取分页总数
```

```
public Object getPageCount()
```

```
{
```

```
    String sqlCmd="SELECT CEIL( COUNT(*)/15.0) FROM V_Card ";
```

```
    return SQLUtil.excuteScalar(sqlCmd, null);
```

```
}
```

```
//根据查询条件获取分页总数
```

```
public Object getPageCountByWhere(String sqlWhere)
```

```
{
```

```
    String sqlCmd="SELECT CEIL( COUNT(*)/15.0) FROM V_Card where  
"+sqlWhere;
```

```
    return SQLUtil.excuteScalar(sqlCmd, null);
```

```
}
```

```
}
```

#### 4.2.2 角色的业务操作

//获取角色表信息列表

```
public List<Object> getEntity()  
{  
    String sqlCmd="select *from Role";  
    return DBUtil.SQLUtil.executeQuery(sqlCmd, null);//执行查询操作  
executeQuery  
}
```

//获取分页后角色表信息列表

```
public List<Object> getEntity(int page)  
{  
    int size=(page-1)*15;  
    String sqlCmd="select * from Role limit "+size+",15";  
    return DBUtil.SQLUtil.executeQuery(sqlCmd, null);//执行查询操作  
executeQuery  
}
```

//根据查询条件sqlWhere获取分页后角色表信息列表

```
public List<Object> getEntityByWhere(String sqlWhere,int page)  
{  
    int size=(page-1)*15;  
    String sqlCmd="select * from Role where "+sqlWhere+" limit "+  
size+",15";  
    System.out.println("sqlCmd =="+sqlCmd);  
    return DBUtil.SQLUtil.executeQuery(sqlCmd, null);//执行查询操作  
executeQuery  
}
```

//删除角色表信息

```
public int deleteEntity(String role_id)  
{  
    String sqlCmd="delete from Role where
```

## 停车场管理系统实验报告

---

```
role_id='"+role_id+"'";
        return DBUtil.SQLUtil.executeNonQuery(sqlCmd, null); //
执行非查询操作executeNonQuery
    }

    //根据角色表编号获取角色表信息
    public List<Object> getEntityById(String role_id)
    {
        String sqlCmd="select *From Role where
role_id='"+role_id+"'";
        System.out.println(sqlCmd);
        return DBUtil.SQLUtil.executeQuery(sqlCmd, null); //执行
查询操作executeQuery
    }

    //根据角色表编号获取角色表信息
    public List<Object> getEntityByName(String role_name)
    {
        String sqlCmd="select * From Role where
role_name='"+role_name+"'";
        System.out.println(sqlCmd);
        return DBUtil.SQLUtil.executeQuery(sqlCmd, null); //执行
查询操作executeQuery
    }

    //更新角色表信息
    public int updateEntity(String role_id,String role_name)
    {
        String sqlCmd="Update Role set role_name='" +
role_name + "' where role_id='"+role_id+"'";
        return SQLUtil.executeNonQuery(sqlCmd, null);
    }

    //插入角色表信息
    public int insertEntity(String role_id,String role_name)
    {
        String sqlCmd="Insert into Role values('" + role_id +
"', '"+role_name+"')";
        return SQLUtil.executeNonQuery(sqlCmd, null);
    }
}
```

```
    }

    //检查插入主键是否重复
    public boolean checkExist(String role_id)
    {
        String sqlCmd="select count(*) from Role where
role_id='"+role_id+"'";
        if(1==Integer.parseInt(SQLUtil.excuteScalar(sqlCmd,
null).toString())) )
        {
            return true;
        }
        return false;
    }

    //获取分页总数
    public Object getPageCount()
    {
        String sqlCmd="SELECT CEIL( COUNT(*)/15.0) FROM Role ";
        return SQLUtil.excuteScalar(sqlCmd, null);
    }

    //根据查询条件获取分页总数
    public Object getPageCountByWhere(String sqlWhere)
    {
        String sqlCmd="SELECT CEIL( COUNT(*)/15.0) FROM Role where
"+sqlWhere;
        return SQLUtil.excuteScalar(sqlCmd, null);
    }

}
```

#### 4. 2. 3 固定车位的逻辑业务

```
//获取固定车主出入记录表信息列表
public List<Object> getEntity()
{
    String sqlCmd="select *from Fixed";
```

## 停车场管理系统实验报告

---

```
        return DBUtil.SQLUtil.executeQuery(sqlCmd, null); //执行查询操作
executeQuery
    }

    //获取分页后固定车主出入记录表信息列表
    public List<Object> getEntity(int page)
    {
        int size=(page-1)*15;
        String sqlCmd="select *from V_Fixed limit "+size+",15";
        return DBUtil.SQLUtil.executeQuery(sqlCmd, null); //执行查询操作
executeQuery
    }

    //获取未出场车辆
    public List<Object> getNoOut(int page)
    {
        int size=(page-1)*15;
        String sqlCmd="select *from V_Fixed where out_date='1111-11-11'
limit "+size+",15";
        return DBUtil.SQLUtil.executeQuery(sqlCmd, null); //执行查询操作
executeQuery
    }

    //根据查询条件sqlWhere获取分页后固定车主出入记录表信息列表
    public List<Object> getEntityByWhere(String sqlWhere,int page)
    {
        int size=(page-1)*15;
        String sqlCmd="select *from V_Fixed where "+sqlWhere+" limit "+
size+",15";
        return DBUtil.SQLUtil.executeQuery(sqlCmd, null); //执行查询操作
executeQuery
    }

    //删除固定车主出入记录表信息
    public int deleteEntity(String fixed_id)
    {
        String sqlCmd="delete from Fixed where
fixed_id='"+fixed_id+"'";
```

## 停车场管理系统实验报告

---

```
        return DBUtil.SQLUtil.executeNonQuery(sqlCmd, null); //执行非查
        询操作executeNonQuery
    }

    //根据固定车主出入记录表编号获取固定车主出入记录表信息
    public List<Object> getEntityById(String fixed_id)
    {
        String sqlCmd="select *From V_Fixed where
        fixed_id='"+fixed_id+"'";
        return DBUtil.SQLUtil.executeQuery(sqlCmd, null); //执行查询操作
        executeQuery
    }

    //更新固定车主出入记录表信息
    public int updateEntity(String fixed_id,String card_id,String
    entry_date,String entry_time,String out_date,String out_time)
    {
        String sqlCmd="Update Fixed set card_id='" + card_id +
        "',entry_date='" + entry_date + "',entry_time='" + entry_time +
        "',out_date='" + out_date + "',out_time='" + out_time + "' where
        fixed_id='"+fixed_id+"'";
        return SQLUtil.executeNonQuery(sqlCmd, null);
    }

    public int setOut(String fixed_id,String out_date,String
    out_time )
    {
        String sqlCmd="update Fixed set
        out_date='"+out_date+"',out_time='"+out_time+"' where
        fixed_id='"+fixed_id+"'";
        return SQLUtil.executeNonQuery(sqlCmd, null);
    }

    //插入固定车主出入记录表信息
    public int insertEntity(String fixed_id,String card_id,String
    entry_date,String entry_time,String out_date,String out_time)
    {
        String sqlCmd="Insert into Fixed values('" + fixed_id + "', '"
```

## 停车场管理系统实验报告

```
+ card_id + "',' + entry_date + "',' + entry_time + "',' + out_date +
 "',''+out_time+'"");
    return SQLUtil.executeNonQuery(sqlCmd, null);
}

//检查插入主键是否重复
public boolean checkExist(String fixed_id)
{
    String sqlCmd="select count(*) from V_Fixed where
fixed_id='"+fixed_id+"'";
    if(1==Integer.parseInt(SQLUtil.excuteScalar(sqlCmd,
null).toString()) )
    {
        return true;
    }
    return false;
}

//获取分页总数
public Object getPageCount()
{
    String sqlCmd="SELECT CEIL( COUNT(*)/15.0) FROM V_Fixed ";
    return SQLUtil.excuteScalar(sqlCmd, null);
}

//根据查询条件获取分页总数
public Object getPageCountByWhere(String sqlWhere)
{
    String sqlCmd="SELECT CEIL( COUNT(*)/15.0) FROM V_Fixed where
"+sqlWhere;
    return SQLUtil.excuteScalar(sqlCmd, null);
}
}
```

### 4. 2. 4 临时车位的逻辑业务

//获取零时车主出入记录表信息列表



## 停车场管理系统实验报告

---

```
public List<Object> getEntity()
{
    String sqlCmd="select *from Temp";
    return DBUtil.SQLUtil.executeQuery(sqlCmd, null);//执行查询操作
executeQuery
}

//获取分页后零时车主出入记录表信息列表
public List<Object> getEntity(int page)
{
    int size=(page-1)*15;
    String sqlCmd="select *from Temp limit "+size+",15";
    return DBUtil.SQLUtil.executeQuery(sqlCmd, null);//执行查询操作
executeQuery
}

//根据查询条件sqlWhere获取分页后零时车主出入记录表信息列表
public List<Object> getEntityByWhere(String sqlWhere,int page)
{
    int size=(page-1)*15;
    String sqlCmd="select *from Temp where "+sqlWhere+" limit "+
size+",15";
    return DBUtil.SQLUtil.executeQuery(sqlCmd, null);//执行查询操作
executeQuery
}

//删除零时车主出入记录表信息
public int deleteEntity(String temp_id)
{
    String sqlCmd="delete from Temp where
temp_id='"+temp_id+"'";
    return DBUtil.SQLUtil.executeNonQuery(sqlCmd, null);//
执行非查询操作executeNonQuery
}

//根据零时车主出入记录表编号获取零时车主出入记录表信息
public List<Object> getEntityById(String temp_id)
{

```

## 停车场管理系统实验报告

---

```
String sqlCmd="select *From Temp where
temp_id='"+temp_id+"'";
return DBUtil.SQLUtil.executeQuery(sqlCmd, null);//执行
查询操作executeQuery
}

//更新零时车主出入记录表信息
public int updateEntity(String temp_id,String
card_id,String car_num,String entry_date,String entry_time,String
out_date,String out_time,String temp_money)
{
String sqlCmd="Update Temp set card_id='" + card_id +
"',car_num='" + car_num + "',entry_date='" + entry_date +
"',entry_time='" + entry_time + "',out_date='" + out_date +
"',out_time='" + out_time + "',temp_money='" + temp_money + "' where
temp_id='"+temp_id+"'";
return SQLUtil.executeNonQuery(sqlCmd, null);
}

//插入零时车主出入记录表信息
public int insertEntity(String temp_id,String
card_id,String car_num,String entry_date,String entry_time,String
out_date,String out_time,String temp_money)
{
String sqlCmd="Insert into Temp values('" + temp_id +
"', '" + card_id + "', '" + car_num + "', '" + entry_date + "', '" +
entry_time + "', '" + out_date + "', '" + out_time + "', '"+temp_money+"')";
return SQLUtil.executeNonQuery(sqlCmd, null);
}

//检查插入主键是否重复
public boolean checkExist(String card_id)
{
String sqlCmd="select count(*) from Temp where
card_id='"+card_id+"'";
if(1==Integer.parseInt(SQLUtil.excuteScalar(sqlCmd,
null).toString())) )
{
return true;
}
```

```
        }
        return false;
    }

    //获取分页总数
    public Object getPageCount()
    {
        String sqlCmd="SELECT CEIL( COUNT(*)/15.0) FROM Temp ";
        return SQLUtil.excuteScalar(sqlCmd, null);
    }

    //根据查询条件获取分页总数
    public Object getPageCountByWhere(String sqlWhere)
    {
        String sqlCmd="SELECT CEIL( COUNT(*)/15.0) FROM Temp where
"+sqlWhere;
        return SQLUtil.excuteScalar(sqlCmd, null);
    }
}

4. 2. 5 车位的逻辑业务
    //获取车位表信息列表
    public List<Object> getEntity()
    {
        String sqlCmd="select *from Seat";
        return DBUtil.SQLUtil.executeQuery(sqlCmd, null);//执行查询操作
    }

    //获取未分配的车位
    public List<Object> getNoUseSeat()
    {
        String sqlCmd="SELECT *FROM Seat WHERE seat_id NOT IN(SELECT
seat_id FROM card)";
        return DBUtil.SQLUtil.executeQuery(sqlCmd, null);//执行查询操作
    }
}
```

## 停车场管理系统实验报告

---

```
executeQuery
```

```
}
```

```
//获取分页后车位表信息列表
```

```
public List<Object> getEntity(int page)
```

```
{
```

```
    int size=(page-1)*15;
```

```
    String sqlCmd="select *from Seat limit "+size+",15";
```

```
    return DBUtil.SQLUtil.executeQuery(sqlCmd, null);//执行查询操作
```

```
executeQuery
```

```
}
```

```
//根据查询条件sqlWhere获取分页后车位表信息列表
```

```
public List<Object> getEntityByWhere(String sqlWhere,int page)
```

```
{
```

```
    int size=(page-1)*15;
```

```
    String sqlCmd="select *from Seat where "+sqlWhere+" limit "+size+",15";
```

```
    System.out.println("sqlWhere="+sqlWhere);
```

```
    return DBUtil.SQLUtil.executeQuery(sqlCmd, null);//执行查询操作
```

```
executeQuery
```

```
}
```

```
//删除车位表信息
```

```
public int deleteEntity(String seat_id)
```

```
{
```

```
    String sqlCmd="delete from Seat where seat_id='"+seat_id+"'";
```

```
    return DBUtil.SQLUtil.executeNonQuery(sqlCmd, null);//执行非查询操作executeNonQuery
```

```
}
```

```
//根据车位表编号获取车位表信息
```

```
public List<Object> getEntityById(String seat_id)
```

```
{
```

```
    String sqlCmd="select *From Seat where seat_id='"+seat_id+"'";
```

```
    return DBUtil.SQLUtil.executeQuery(sqlCmd, null);//执行查询操作
```

```
executeQuery
```

## 停车场管理系统实验报告

---

```
}

//更新车位表信息
public int updateEntity(String seat_id,String seat_num,String
seat_section,String seat_state,String seat_tag)
{
    String sqlCmd="Update Seat set seat_num='" + seat_num +
    "',seat_section='" + seat_section + "',seat_state='" + seat_state +
    "',seat_tag='" + seat_tag + "' where seat_id='"+seat_id+"'";
    return SQLUtil.executeNonQuery(sqlCmd, null);
}

//插入车位表信息
public int insertEntity(String seat_id,String seat_num,String
seat_section,String seat_state,String seat_tag)
{
    String sqlCmd="Insert into Seat values('" + seat_id + "',''" +
seat_num + "',''" + seat_section + "',''" + seat_state +
    "','"+seat_tag+"')";
    return SQLUtil.executeNonQuery(sqlCmd, null);
}

//检查插入主键是否重复
public boolean checkExist(String seat_id)
{
    String sqlCmd="select count(*) from Seat where
seat_id='"+seat_id+"'";
    if(1==Integer.parseInt(SQLUtil.excuteScalar(sqlCmd,
null).toString())) )
    {
        return true;
    }
    return false;
}

//获取分页总数
public Object getPageCount()
{
```

```
String sqlCmd="SELECT CEIL( COUNT(*)/15.0) FROM Seat ";
return SQLUtil.excuteScalar(sqlCmd, null);
}

//根据查询条件获取分页总数
public Object getPageCountByWhere(String sqlWhere)
{
    String sqlCmd="SELECT CEIL( COUNT(*)/15.0) FROM Seat where
"+sqlWhere;
    return SQLUtil.excuteScalar(sqlCmd, null);
}

}
```

### 4.3 创建 Servlet 层

Servlet 相当于一个构造器，将前端和后端的数据通过 Servlet 流通联系起来，通过重写其中的 doGet() 和 doPost() 方法接受并处理得到数据，它必须继承 HttpServlet。下面介绍几个比较重要的部分：

#### 4.3.1 固定车位的数据处理（增删改查）

项目成果展示图：

The screenshot displays a web application interface for a parking management system. It features two main sections: 'Add Entry Information' (添加入场信息) and 'Exit Information Management' (出场信息管理). The 'Add Entry Information' section includes a dropdown menu for 'IC Card Number' (IC卡号) with the value '20200521190631(李小龙)' and a 'Confirm' (确定) button. The 'Exit Information Management' section includes a 'Query Condition' (查询条件) dropdown set to 'Record Number' (记录编号), a 'Query Value' (查询值) input field, and a 'Query' (查询) button. Below these sections is a table with columns: 'Record Number' (记录编号), 'IC Card' (IC卡号), 'Vehicle Name' (车主名称), 'Vehicle License Plate' (车牌号码), 'Entry Date' (入场日期), 'Exit Status' (是否出场), and 'Action' (操作). At the bottom right of the table, there is a pagination control showing 'Total 0 pages' (共 0 页) and buttons for 'Jump to' (跳转至) and 'Go' (转).

//通过表单get方式传值 将进入doGet函数 (method="get")

```
public void doGet(HttpServletRequest request,
HttpServletResponse response)
    throws ServletException, IOException {
    this.response=response;
    this.request=request;
    int
handleType=Integer.parseInt(request.getParameter("type").toString());
    switch (handleType) {
        case 1://类型1代表删除表中的数据
            deleteEntity();
```

## 停车场管理系统实验报告

---

```
        break;
    case 4://类型4代表获取表中信息
        getEntity();
        break;
    case 5://类型5代表根据查询条件获取表中信息
        getEntityByWhere();
        break;
    case 6://类型6代表管理员获取未出场车辆
        getNoOut();
        break;
    case 10://类型10代表更新车辆出场
        setOut();
        break;
    default:
        break;
    }
}
```

//通过表单post方式传值 将进入doPost函数（method="post"）

```
public void doPost(HttpServletRequest request,
HttpServletResponse response)
    throws ServletException, IOException {
    this.request=request;
    this.response=response;
    int
handleType=Integer.parseInt(request.getParameter("type").toString());//将
前台页面传过来的type类型转化成整型
    switch (handleType) {
    case 2://类型2代表更新表中的数据
        updateEntity();
        break;
    case 3://类型3代表向表中添加数据
        insertEntity();
        break;
    default:
        break;
    }
}
```

## 停车场管理系统实验报告

---

```
    }  
    //删除数据操作  
    private void deleteEntity() throws IOException  
    {  
        String fixed_id=request.getParameter("fixed_id");//获取前台通  
过get方式传过来的JId  
        fixed.deleteEntity(fixed_id);//执行删除操作  
        response.sendRedirect("/Parking/FixedHandle?type=4");//删除  
成功后跳转至管理页面  
    }  
  
    //车辆出场更新操作  
    private void setOut() throws IOException  
    {  
        String fixed_id=new  
String(request.getParameter("fixed_id").getBytes("ISO8859_1"),"UTF-8");  
        SimpleDateFormat dateFormat =new    SimpleDateFormat("yyyy-  
MM-dd");  
        String out_date=dateFormat.format(new Date());  
        SimpleDateFormat timeFormat =new  
SimpleDateFormat("HH:mm:ss");  
        String out_time=timeFormat.format(new Date());  
        if(fixed.setOut(fixed_id, out_date, out_time)==1)  
        {  
            response.sendRedirect("/Parking/FixedHandle?type=6");  
        }  
    }  
  
    //更新数据操作  
    private void updateEntity() throws UnsupportedEncodingException  
    {  
        String fixed_id=new  
String(request.getParameter("fixed_id").getBytes("ISO8859_1"),"UTF-8");  
        String card_id=new  
String(request.getParameter("card_id").getBytes("ISO8859_1"),"UTF-8");  
        String entry_date=new  
String(request.getParameter("entry_date").getBytes("ISO8859_1"),"UTF-8");  
        String entry_time=new
```



## 停车场管理系统实验报告

```
String(request.getParameter("entry_time").getBytes("ISO8859_1"), "UTF-8");
    String out_date=new
String(request.getParameter("out_date").getBytes("ISO8859_1"), "UTF-8");
    String out_time=new
String(request.getParameter("out_time").getBytes("ISO8859_1"), "UTF-8");

    if(fixed.updateEntity(fixed_id,card_id,entry_date,entry_time,out_date
,out_time)==1)
    {
        try {

            response.sendRedirect("/Parking/FixedHandle?type=4");//成功更新数据后跳
转至FixedMsg.jsp页面
        } catch (IOException e) {
            e.printStackTrace();//异常处理
        }
    }

    //插入数据操作
    private void insertEntity() throws
UnsupportedEncodingException, IOException
    {
        response.setCharacterEncoding("UTF-8");
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out=response.getWriter();

        SimpleDateFormat dateFormat =new
SimpleDateFormat("yyyyMMddHHmmss");
        String fixed_id=dateFormat.format(new Date());
        String card_id=new
String(request.getParameter("card_id").getBytes("ISO8859_1"), "UTF-8");
        SimpleDateFormat dFormat =new    SimpleDateFormat("yyyy-MM-
dd");
        String entry_date=dFormat.format(new Date());
        SimpleDateFormat tFormat =new
SimpleDateFormat("HH:mm:ss");
```

## 停车场管理系统实验报告

---

```
String entry_time=tFormat.format(new Date());
String out_date="1111-11-11";
String out_time="11:11:11";

if(!fixed.checkExist(fixed_id))
{

    if(fixed.insertEntity(fixed_id,card_id,entry_date,entry_time,out_date
,out_time)==1)
    {
        out.write("<script>alert('数据添加成功! ');");
location.href = '/Parking/FixedHandle?type=6';</script>");
    }
    else {
        out.write("<script>alert('数据添失败! '); location.href
= '/Parking/FixedHandle?type=6';</script>");
    }
}
else {
    out.write("<script>alert('主键重复，数据添加失败! ');");
location.href = '/Parking/FixedHandle?type=4';</script>");
}
}

//获取对象所有数据列表
private void getEntity() throws ServletException, IOException
{
    request.setCharacterEncoding("UTF-8");
    int
page=request.getParameter("page")==null?1:Integer.parseInt(request.getPar
ameter("page").toString());//获取跳转的页面号
    int
totalPage=Integer.parseInt(fixed.getPageCount().toString()) ;//获取分页总
数

    List<Object> list=fixed.getEntity(page);//获取数据列表
    request.setAttribute("list",list);//将数据存放到request对象
中，用于转发给前台页面使用
    request.setAttribute("totalPage",totalPage );//将totalPage存
```

## 停车场管理系统实验报告

---

放到request对象中，用于转发给前台页面使用

```
request.getRequestDispatcher("/Admin/FixedMsg.jsp").forward(request,
response);//请求转发
}

//获取未出场的车辆
private void getNoOut() throws ServletException, IOException
{
    request.setCharacterEncoding("UTF-8");
    int
page=request.getParameter("page")==null?1:Integer.parseInt(request.getPar
ameter("page").toString());//获取跳转的页面号
    int
totalPage=Integer.parseInt(fixed.getPageCount().toString()) ;//获取分页总
数
    List<Object> list=fixed.getNoOut(page);//获取数据列表
    request.setAttribute("list",list);//将数据存放到request对象
中，用于转发给前台页面使用
    request.setAttribute("totalPage",totalPage );//将totalPage存
放到request对象中，用于转发给前台页面使用

    request.getRequestDispatcher("/Admin/FixedOut.jsp").forward(request,
response);//请求转发
}

//根据查询条件获取对象所有数据列表
private void getEntityByWhere() throws ServletException,
IOException
{
    request.setCharacterEncoding("UTF-8");
    String condition=request.getParameter("condition");//获取查询
字段的名称
    //String value=new
String(request.getParameter("value").getBytes("ISO8859_1"),"UTF-8");//获
取查询的值
    String value = request.getParameter("value");
    String where=condition+"=\""+value+"\"";//拼接查询字符串
    int
```

## 停车场管理系统实验报告

```
page=request.getParameter("page")==null?1:Integer.parseInt(request.getParameter("page")); //获取要跳转的页面号
    int
wherePage=Integer.parseInt(fixed.getPageCountByWhere(where).toString()); //获取查询后的分页总数
    List<Object> list=fixed.getEntityByWhere(where, page); //获取查询后的数据列表
    request.setAttribute("list",list); //将数据存放到request对象中，用于转发给前台页面使用
    request.setAttribute("wherePage",wherePage);
    request.setAttribute("condition",condition);
    request.setAttribute("value",value);

    request.getRequestDispatcher("/Admin/FixedMsg.jsp").forward(request, response);
}
}
```

### 4.3.2 临时车位数据处理

项目成果展示图;

零时IC卡号:

车牌号码:

查询条件: 

车牌号码

 查询值: 

查询

零时编号	零时IC卡号	车牌号码	入场日期	出场日期	停车费用	操作
20200925173007	川B23333	川B23333	2020-09-25 17:30:07	2020-11-28 21:29:26	20.0	删除 打印
20200925203021	川B23333	川B23333	2020-09-25 20:30:21	2020-11-28 21:29:26	15.0	删除 打印
20201201190239	川B11111	川B11111	2020-12-01 19:02:39	2020-12-01 19:04:24	3.0	删除 打印
20201201190418	川F22222	川F22222	2020-12-01 19:04:18	2020-12-01 19:04:28	3.0	删除 打印

共 1 页 跳转至 

转

```
public class TempHandle extends HttpServlet {
```

```
    HttpServletRequest request;
```

## 停车场管理系统实验报告

---

```
HttpServletResponse response;
Service.Temp temp=new Service.Temp();

//通过表单get方式传值 将进入doGet函数（method="get"）
public void doGet(HttpServletRequest request,
HttpServletResponse response)
    throws ServletException, IOException {
    this.response=response;
    this.request=request;
    int
handleType=Integer.parseInt(request.getParameter("type").toString());
    switch (handleType) {
        case 1://类型1代表删除表中的数据
            deleteEntity();
            break;
        case 4://类型4代表获取表中信息
            getEntity();
            break;
        case 5://类型5代表根据查询条件获取表中信息
            getEntityByWhere();
            break;
        default:
            break;
    }
}

//通过表单post方式传值 将进入doPost函数（method="post"）
public void doPost(HttpServletRequest request,
HttpServletResponse response)
    throws ServletException, IOException {
    this.request=request;
    this.response=response;
    int
handleType=Integer.parseInt(request.getParameter("type").toString());//将
前台页面传过来的type类型转化成整型
    switch (handleType) {
        case 2://类型2代表更新表中的数据
```

## 停车场管理系统实验报告

---

```
        updateEntity();
        break;
    case 3://类型3代表向表中添加数据
        insertEntity();
        break;
    default:
        break;
    }
}

//删除数据操作
private void deleteEntity() throws IOException
{
    String temp_id=request.getParameter("temp_id");//获取前台通过
get方式传过来的JId
    temp.deleteEntity(temp_id);//执行删除操作
    response.sendRedirect("/Parking/TempHandle?type=4");//删除成
功后跳转至管理页面
}

//更新数据操作
private void updateEntity() throws UnsupportedEncodingException
{
    String temp_id=new
String(request.getParameter("temp_id").getBytes("ISO8859_1"),"UTF-8");
String card_id=new
String(request.getParameter("card_id").getBytes("ISO8859_1"),"UTF-8");
String car_num=new
String(request.getParameter("car_num").getBytes("ISO8859_1"),"UTF-8");
String entry_date=new
String(request.getParameter("entry_date").getBytes("ISO8859_1"),"UTF-8");
String entry_time=new
String(request.getParameter("entry_time").getBytes("ISO8859_1"),"UTF-8");
String out_date=new
String(request.getParameter("out_date").getBytes("ISO8859_1"),"UTF-8");
String out_time=new
String(request.getParameter("out_time").getBytes("ISO8859_1"),"UTF-8");
String temp_money=new
```

## 停车场管理系统实验报告

```
String(request.getParameter("temp_money").getBytes("ISO8859_1"), "UTF-8");

    if(temp.updateEntity(temp_id,card_id,car_num,entry_date,entry_time,out_date,out_time,temp_money)==1)
    {
        try {

            response.sendRedirect("/Parking/TempHandle?type=4");//成功更新数据后跳转至TempMsg.jsp页面
        } catch (IOException e) {
            e.printStackTrace();//异常处理
        }
    }
}
//插入数据操作
private void insertEntity() throws
UnsupportedEncodingException, IOException
{
    response.setCharacterEncoding("UTF-8");
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out=response.getWriter();

    SimpleDateFormat dateFormat =new
SimpleDateFormat("yyyyMMddHHmmss");
    String temp_id=dateFormat.format(new Date());
    //String card_id=new
String(request.getParameter("card_id").getBytes("ISO8859_1"), "UTF-8");
    //String car_num=new
String(request.getParameter("car_num").getBytes("ISO8859_1"), "UTF-8");
    String card_id = request.getParameter("card_id");
    String car_num = request.getParameter("car_num");
    SimpleDateFormat dFormat =new SimpleDateFormat("yyyy-MM-
dd");
    String entry_date=dFormat.format(new Date());
    SimpleDateFormat tFormat =new
SimpleDateFormat("HH:mm:ss");
    String entry_time=tFormat.format(new Date());
```

## 停车场管理系统实验报告

---

```
String out_date=null;
String out_time=null;
String temp_money="0";

if(!temp.checkExist(card_id))
{
    System.out.println("11111");
    if((card_id!=null
&&card_id!="")&&(car_num!=null&&car_num!="")){

        if(temp.insertEntity(temp_id,card_id,car_num,entry_date,entry_time,out_date,out_time,temp_money)==1)
        {
            System.out.println("card_id="+card_id);
            System.out.println("car_num="+car_num);
            out.write("<script>alert('数据添加成功! ');
location.href = '/Parking/TempHandle?type=4';</script>");
        }
        else {
            out.write("<script>alert('数据添失败! '); location.href
= '/Parking/TempHandle?type=4';</script>");
        }
    }
    else{
        out.write("<script>alert('临时IC卡号或者车牌号不能为空
'); location.href = '/Parking/TempHandle?type=4';</script>");
    }

}

else {
    out.write("<script>alert('主键重复，数据添加失败! ');
location.href = '/Parking/TempHandle?type=4';</script>");
}
}

//获取对象所有数据列表
private void getEntity() throws ServletException, IOException
{
    request.setCharacterEncoding("UTF-8");
```



## 停车场管理系统实验报告

---

```
        int
page=request.getParameter("page")==null?1:Integer.parseInt(request.getParameter("page").toString()); //获取跳转的页面号
        int
totalPage=Integer.parseInt(temp.getPageCount().toString()) ; //获取分页总数
        List<Object> list=temp.getEntity(page); //获取数据列表
        request.setAttribute("list",list); //将数据存放到request对象
中，用于转发给前台页面使用
        request.setAttribute("totalPage",totalPage ); //将totalPage存
放到request对象中，用于转发给前台页面使用

        request.getRequestDispatcher("/Admin/TempMsg.jsp").forward(request,
response); //请求转发
    }

    //根据查询条件获取对象所有数据列表
    private void getEntityByWhere() throws ServletException,
IOException
    {
        request.setCharacterEncoding("UTF-8");
        String condition=request.getParameter("condition"); //获取查询
字段的名称
        //String value=new
String(request.getParameter("value").getBytes("ISO8859_1"),"UTF-8"); //获
取查询的值
        String value = request.getParameter("value");
        String where=condition+"=\""+value+"\""; //拼接查询字符串
        int
page=request.getParameter("page")==null?1:Integer.parseInt(request.getParameter("page")); //获取要跳转的页面号
        int
wherePage=Integer.parseInt(temp.getPageCountByWhere(where).toString()) ; //
/获取查询后的分页总数
        List<Object> list=temp.getEntityByWhere(where, page); //获取
查询后的数据列表
        request.setAttribute("list",list); //将数据存放到request对象
中，用于转发给前台页面使用
        request.setAttribute("wherePage",wherePage );
        request.setAttribute("condition",condition);
```

## 停车场管理系统实验报告

```
request.setAttribute("value",value);
```

```
request.getRequestDispatcher("/Admin/TempMsg.jsp").forward(request,
response);
    }
}
```

### 4.4 页面设计

页面设计这一块使用了 HTML 的一些基本内容其中用 JSP 嵌套了 Java 代码处理数据，CSS 作了 Style 页面风格处理，然后在侧栏的滑动效果使用了 JS 技术引入了 JQuery。整体符合基本网页结构。

#### 4.4.1 登录页面设计

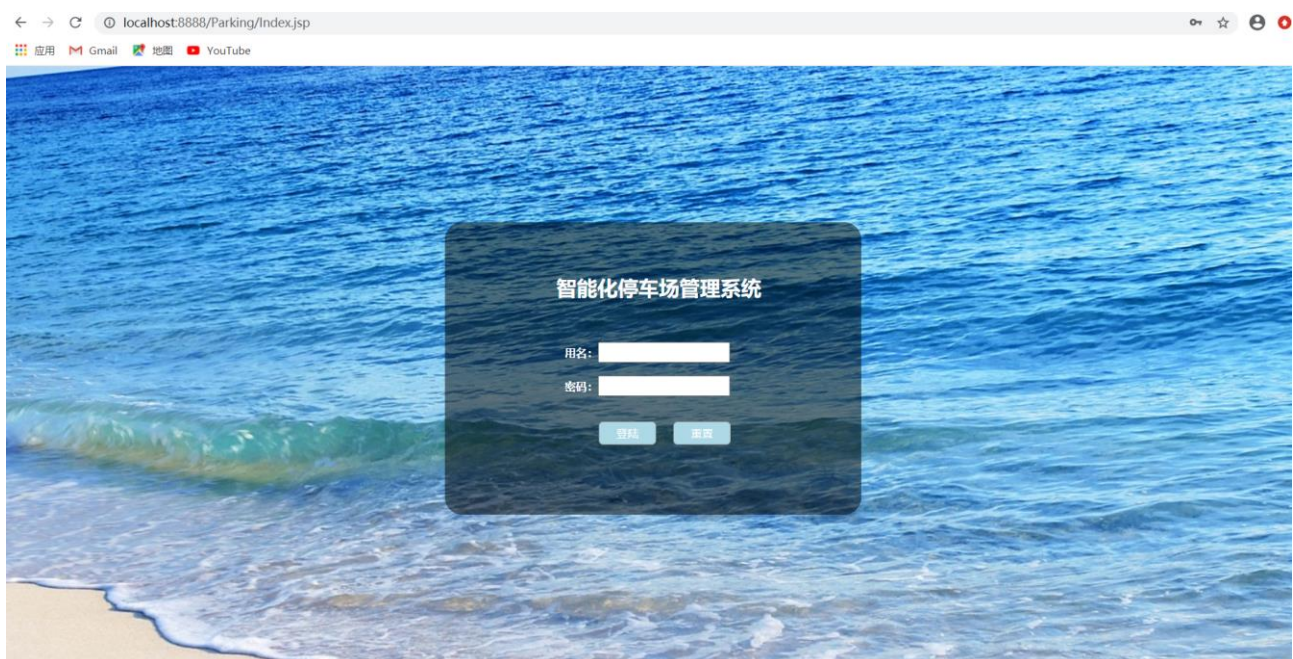


图 4.3.1 登录页面设计

关键代码如下：

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%
    String path = request.getContextPath();//获取项目名称
%>
```

## 停车场管理系统实验报告

---

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>欢迎登陆</title>
<link rel="stylesheet" type="text/css" href="<%=path %>/Style/Login.css"
/>
<script type="text/javascript" src="Script/jquery-1.10.1.js"></script>
<script type="text/javascript">
    $(function()
    {
        $(document).on("click", ".a_reg", function()
        {
            $(".model").fadeIn();
        });

        $(document).on("click", ".a_close", function()
        {
            $(".model").fadeOut();
        });
    });

    function checkPwd()
    {
        if($(".name=user_pwd1").val() != $(".name=re_pwd").val())
        {
            alert("两次输入密码不一致~~~");
            return false;
        }
        else
        {
            return true;
        }
    };
</script>
</head>
<body>
```

## 停车场管理系统实验报告

```
<div class="wrapLogin">
    <div class="loginPanel">
        <form action="<%=path %>/LoginHandle" method="post">
            <h2>智能化停车场管理系统</h2>
            <p><label>用户名: </label><input type="text"
name="user_id" value="SAdmin" /></p>
            <p><label>密码: </label><input type="password"
name="user_pwd" value="123123" /></p>
            <p class="btn"><input type="submit" class="btnLogin"
value="登陆" /><input type="button" class="btnCancel" value="重置"
/></p>
        </form>
    </div>
</div>
<!-- loginPanel End -->
</body>
</html>
```

### 4.4.2 主页面设计



图 4.3.2 停车场管理系统主页面设计

关键代码如下：

## 停车场管理系统实验报告

---

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8" errorPage="_Error.jsp"%>
<%
    String path = request.getContextPath();//获取项目名称
%>
<!doctype html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>智能化停车场管理系统</title>
<link href="<%=path %>/Style/Index.css" rel="stylesheet" type="text/css"
/>
<script type="text/javascript" src="<%=path %>/Script/jquery-
1.10.1.js"></script>
<script type="text/javascript" src="<%=path %>/Script/Index.js"></script>
</head>
<body>

    <!-- 判断uName是否存在，如何不存在则证明非登录操作，跳转至登录页面 -->
    <% if (session.getAttribute("user_name") == null ) {%>
    <jsp:forward page="Login.jsp" ></jsp:forward>
    <% } %>

    <!--header-->
    <div class="header">
        <div class="header_logo">
            智能化停车场管理系统
        </div>
    <!--
```

## 停车场管理系统实验报告

---

```
<div class="func">
    <ul>
        <li class="li_func1"><a
href="<%=path %>/Common/ChagePwd.jsp" target="mainFrame">修改密码
</a></li>
        <li class="li_func2"><a
href="<%=path %>/Common/UserInfo.jsp" target="mainFrame">用户信息
</a></li>
        <li class="li_func3"><a
href="<%=path %>/Common/Logout.jsp">退出系统</a></li>
    </ul>
</div>
--%>
</div>
```

```
<div class="wrap">
    <ul class="siderbar">

        <%

if(session.getAttribute("role_id").toString().equals("r001"))
    {
        %>
        <li><span>系统信息管理</span>
        <ul>

            <li><a href="<%=path %>/Admin/RoleAdd.jsp"
target="mainFrame">添加角色信息</a></li>
            <li><a href="<%=path %>/RoleHandle?type=4"
target="mainFrame">管理角色信息</a></li>
```

## 停车场管理系统实验报告

---

```
        <li><a href="<%=path %>/Admin/UserAdd.jsp"
target="mainFrame">添加用户信息</a></li>

        <li><a href="<%=path %>/UserHandle?type=4"
target="mainFrame">管理用户信息</a></li>

    </ul>

</li>

<li><span>车位信息管理</span>

    <ul>

        <li><a href="<%=path %>/Admin/SeatAdd.jsp"
target="mainFrame">添加车位信息</a></li>

        <li><a href="<%=path %>/SeatHandle?type=4"
target="mainFrame">管理车位信息</a></li>

    </ul>

</li>

<li><span>IC卡信息管理</span>

    <ul>

        <li><a href="<%=path %>/Admin/CardAdd.jsp"
target="mainFrame">添加IC卡类型</a></li>

        <li><a href="<%=path %>/CardHandle?type=4"
target="mainFrame">管理IC卡类型</a></li>

    </ul>

</li>

<li><span>固定车主停车管理</span>

    <ul>

        <li><a href="<%=path %>/FixedHandle?type=6"
target="mainFrame">出入口设置</a></li>

        <li><a href="<%=path %>/FixedHandle?type=4"
target="mainFrame">停车信息管理</a></li>

    </ul>

</li>

<li><span>临时车辆停车管理</span>
```

## 停车场管理系统实验报告

---

```
<ul>

    <li><a href="<%=path %>/Admin/TempAdd.jsp"
target="mainFrame">车主入场信息</a></li>

    <li><a href="<%=path %>/TempHandle?type=4"
target="mainFrame">车主出场信息</a></li>

</ul>

</li>

<li><span>系统功能操作</span>

    <ul>

        <li><a href="<%=path %>/Common/ChagePwd.jsp"
target="mainFrame">修改密码</a></li>

        <li><a href="<%=path %>/Common/Logout.jsp">退出系统
</a></li>

    </ul>

</li>

<%
}
else
{
%>

<p>留言信息管理</p>

<div>

    <a href="<%=path %>/User/MsgAdd.jsp"
target="mainFrame">添加留言信息</a>

    <a href="<%=path %>/MsgHandle?type=4"
target="mainFrame">管理留言信息</a>

</div>

<li><span>系统功能操作</span>

<div>
```



```
        <a href="<%=path %>/Common/ChagePwd.jsp"
target="mainFrame">修改密码</a>

        <a href="<%=path %>/Common/UserInfo.jsp"
target="mainFrame">个人信息</a>

        <a href="<%=path %>/Common/Logout.jsp">退出系统</a>

    </div>

    <%
    }
    %>

</ul>

    <div class="content">

        <iframe width="99%" height="100%" name="mainFrame"
frameborder="0" >

        </iframe>

    </div>
</div>

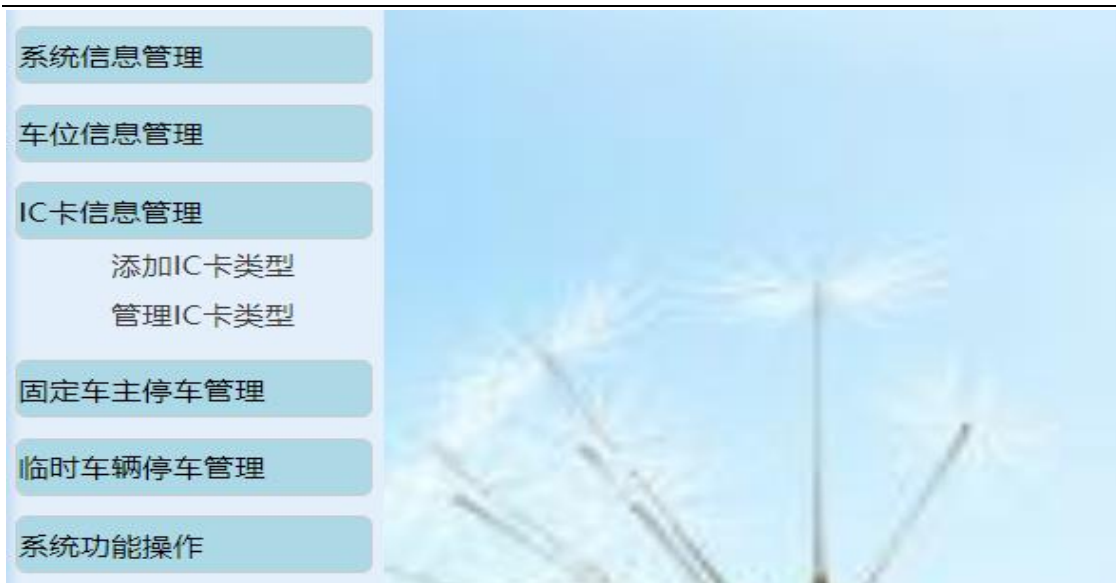
<div class="footer"></div>
</body>
</html>
```

#### 4.4.3 滑动效果设计

鼠标点击侧栏框上，子内容会有一个滑动效果，展示如下：

## 停车场管理系统实验报告

---



JS 关键代码如下：

```
// JavaScript Document

$(function()
{
    autoHeight();

    $(window).resize(function()
    {
        autoHeight();
    });

    $(".siderbar p").click(function()
    {
        if($(this).next("div").css("display")!="block")
        {
            $(".siderbar div").slideUp();
            $(this).next("div").slideDown();
        }
    });
});
```

```
$(".siderbar a").click(function()
{
    $(".siderbar a").css({"color":"#333","font-
weight":"normal","background":"none"});
    $(this).css({"color":"#fff","font-
weight":"bold","background":"#368D81"});
});
//滑动效果
$(".siderbar>li>ul").slideUp();
$(".siderbar>li>span").on("click",function(){
    $(".siderbar>li>ul").slideUp();
    $(this).next().slideDown();
});
$(".siderbar>li>span").eq(0).trigger("click");
});

function autoHeight()
{
    var lessHeight= $(window).height()-$(".header").outerHeight()-
$(".footer").outerHeight();
    $(".siderbar,.content").height(lessHeight-8);
}
```

## 五、 心得体会

在小组接近一个月学习 JSP 和一些前端的相关知识后，深刻地体会到了一起学习分工开发项目的乐趣，不管是在学习新知识的过程中还是在修改报错的过程中，能力都得到了很大的提升，最重要的是每个人都找到了合适自己的学习方法，也找到了自己的定位，有的向后端，也有的人前端也做的非常好，希望以后可以在自己的方向上更上一层楼，

也希望以后有更多的合作机会。当然做的不好的地方还很多,这也需要日后花费更多的时间去提升自己,同时也要帮助团队共同进步。

## 六、参考文献

[1]陈承欢。html5+css3 网页设计与制作实用教程[M].北京:人民邮电出版社,2018.

[3]孔志文。HTML 与 CSS3 技术在网页制作中的应用及发展前景[J].课程教育研究,2015