**摘 要**

竞速用无人机 (Racing Drone)，我们一般称之为穿越机。它与我们日常见到的航拍机完全不同：航拍机是为了拍摄，所以它追求的是稳定、安全和拍摄能力;但是穿越机追求的只有速度、反应和灵活。 为了在赛道上成为最快的无人机，穿越机一般是 DIY 的天下。与组装电脑相类似，组装一台竞速无人机也要去逐一挑选配件，这其中会接触到很多专业的设备性能参数。这些参数并不是每个玩家都可以理解的，而且各个厂家由于没有统一的规范所以使用的参数在单位等方面都有较大的出入，很不便于新手及没有相关专业知识的玩家上手。



竞速无人机在线资料库就在这种背景下应运而生。本程序旨在为广大竞速无人机爱好者提供专业，统一规范，精准的竞速无人机相关产品的参数，性能分析，使用手册等信息，并提供在线虚拟装机功能以及玩家评论等附加功能。本程序为传统BS应用程序，并针对手机和电脑访问做了相应优化，使其在手机和电脑上拥有相同的用户体验。程序使用灵活，可按照电机，电子调速器，飞行控制器等多种分类进行分类浏览，也可对数据库进行模糊查找。本程序使用了java web的技术体系，经过测试和实际使用，各项指标均达到设计目标。

本文针对无人机玩家的需求，具体阐述了竞速无人机在线资料库的设计、实现、部署及测试。目前本程序以通过阿里云平台上线（www.racingdronewiki.com）并初步开始服务。

1绪论

# 1.1课题背景

随着无人机的发展，无人机领域已经得到了飞速的发展，穿越机就算是无人机家族中一特个立独行的款式，有些会玩的玩家通常都会自己买配件进行组装，(就像当年的四驱车一样)，他们会自己组建战队，拍摄自己的飞行视频，追逐各种竞速比赛，其中在 各种穿越机比赛获得冠军就成为了国内众多玩家的追求。

2017竞速无人机（穿越机）走进奥林匹克赛场自去年以来，无人机驾驶竞赛人气飙升，在迪拜举办的奖金100万美元的世界无人机大奖赛是迄今为止最令人瞩目的大赛之一。

迪拜的世界无人机竞速赛才刚刚落下帷幕，近日又有外媒报道称，阿联酋赛事举办方高层还宣布将在2017年12月召开世界未来体育竞赛（World Future Sports Games），比赛为期三天，将每隔两年举办一次。比赛的项目包括：竞速无人机穿越飞行、器人游泳、跑步、摔跤赛事等，可以说是机器人界的奥林匹克运动会。

# 1.2相关研究

互联网技术的飞速发展带动了竞速无人机的高速发展，如今的竞速无人机并不像它的前辈航模一样主要依靠言传身教等线下交流方式。竞速无人机所使用的大量技术都是由一些开源机构、开源社区来进行维护的，且国内目前已出现了一些氛围好、人员专业的论坛和社区。下面来介绍一些其中的佼佼者。

### 1.2.1BetaFlight

BetaFlight是用于多旋翼飞行器和固定翼飞行器的飞行控制器软件（固件）。

Betaflight具有以下功能：

* 多色RGB LED条支持（每个LED可以是不同颜色使用可变长度WS2811 RGB条 - 用于方位指示，低电量警告，飞行模式状态，初始化，故障排除等）
* DShot（150,300,600和1200），Multishot和Oneshot（125和42）电机协议支持
* Blackbox飞行记录仪记录（到板载闪存或外部microSD卡）
* 支持使用STM32 F7，F4，F3和F1处理器的飞行控制器
* PWM，PPM和串行（SBus，SumH，SumD，Spektrum 1024/2048，XBus等）具有故障安全检测的RX连接
* 多个遥测协议（CSRF，FrSky，HoTT smart-port，MSP等）
* RSSI通过ADC - 使用ADC读取PWM RSSI信号，使用FrSky D4R-II，X8R，X4R-SB和XSR进行测试
* OSD支持和配置，无需第三方OSD软件/固件/通讯设备
* OLED显示器 - 显示信息：电池电压/电流/ mAh，配置文件，速率配置文件，模式，版本，传感器等
* 飞行中手动PID调节和速率调整
* 价格简介和飞行中的选择
* 串行接收，遥测，ESC遥测，MSP，GPS，OSD，声纳等可配置的串行端口 - 使用大多数设备在任何端口，软件上运行
* VTX支持Unify Pro和IRC Tramp
* 和更多，更多。

### 1.2.2

模友之吧（http://www.moz8.com）于2013年创建的公益性学习网站！建站初衷是为国内航模无人机及其他类模型爱好者创造一个更具团结、互助、开源、创新品质的精神园地；论坛成立以来，通过测试、捐助、举办设计比赛等多种形式免费赠送给会员各类航模、无人机配件、耗材、图纸等10000多人次，价值近100万元，为模型界培养了大量的后备力量，论坛积累了大量优秀的学习资料和教程550000多篇，目前共有全国（包括港澳台）会员63000多人，未来两年内预计可以突破100000会员，力争在10年内秉持初衷，成为全球最好的模型爱好者综合性中文交流平台。

2相关技术概况

在介绍竞速无人机在线数据库之前，先对本软件的架构以及技术原理进行简要介绍。

# 2.1 Java Web

Java Web，是用Java技术来解决相关web互联网领域的技术总和。web包括：web服务器和web客户端两部分。Java在客户端的应用有java applet，不过使用得很少，Java在服务器端的应用非常的丰富，比如Servlet，JSP和第三方框架等等。Java技术对Web领域的发展注入了强大的动力。

### 2.1.1 Java语言

Java是一门面向对象编程语言，不仅吸收了C++语言的各种优点，还摒弃了C++里难以理解的多继承、指针等概念，因此Java语言具有功能强大和简单易用两个特征。Java语言作为静态面向对象编程语言的代表，极好地实现了面向对象理论，允许程序员以优雅的思维方式进行复杂的编程。

Java具有简单性、面向对象、分布式、健壮性、安全性、平台独立与可移植性、多线程、动态性等特点。Java可以编写桌面应用程序、Web应用程序、分布式系统和嵌入式系统应用程序等。

### 2.1.2 Spring

Spring是一个开放源代码的设计层面框架，他解决的是业务逻辑层和其他各层的松耦合问题，因此它将面向接口的编程思想贯穿整个系统应用。Spring是于2003 年兴起的一个轻量级的Java 开发框架，由Rod Johnson创建。简单来说，Spring是一个分层的Java SE/EE full-stack(一站式) 轻量级开源框架。

### 2.1.3 Spring MVC

通过策略接口，Spring 框架是高度可配置的，而且包含多种视图技术，例如 Java Server Pages（JSP）技术、Velocity、Tiles、iText和POI。Spring MVC 框架并不知道使用的视图，所以不会强迫开发者只使用 JSP 技术。Spring MVC 分离了控制器、模型对象、过滤器以及处理程序对象的角色，这种分离让它们更容易进行定制。

### 2.1.4 Servlet

Servlet（Server Applet）是Java Servlet的简称，称为小服务程序或服务连接器，用Java编写的服务器端程序，主要功能在于交互式地浏览和修改数据，生成动态Web内容。

Servlet运行于支持Java的应用服务器中。从原理上讲，Servlet可以响应任何类型的请求，但绝大多数情况下Servlet只用来扩展基于HTTP协议的Web服务器。

最早支持Servlet标准的是JavaSoft的Java Web Server，此后，一些其它的基于Java的Web服务器开始支持标准的Servlet。

### 2.1.5 Jsp脚本语言

JSP全名为Java Server Pages，中文名叫java服务器页面，其根本是一个简化的Servlet设计，它是由Sun Microsystems公司倡导、许多公司参与一起建立的一种动态网页技术标准。JSP技术有点类似ASP技术，它是在传统的网页HTML（标准通用标记语言的子集）文件(\*.htm,\*.html)中插入Java程序段(Scriptlet)和JSP标记(tag)，从而形成JSP文件，后缀名为(\*.jsp)。 用JSP开发的Web应用是跨平台的，既能在Linux下运行，也能在其他操作系统上运行。

它实现了Html语法中的java扩展（以 <%, %>形式）。JSP与Servlet一样，是在服务器端执行的。通常返回给客户端的就是一个HTML文本，因此客户端只要有浏览器就能浏览。

JSP技术使用Java编程语言编写类XML的tags和scriptlets，来封装产生动态网页的处理逻辑。网页还能通过tags和scriptlets访问存在于服务端的资源的应用逻辑。JSP将网页逻辑与网页设计的显示分离，支持可重用的基于组件的设计，使基于Web的应用程序的开发变得迅速和容易。 JSP(JavaServer Pages)是一种动态页面技术，它的主要目的是将表示逻辑从Servlet中分离出来。

Java Servlet是JSP的技术基础，而且大型的Web应用程序的开发需要Java Servlet和JSP配合才能完成。JSP具备了Java技术的简单易用，完全的面向对象，具有平台无关性且安全可靠，主要面向因特网的所有特点。

# 2.2 Tomcat 服务器

Tomcat是Apache 软件基金会（Apache Software Foundation）的Jakarta 项目中的一个核心项目，由Apache、Sun 和其他一些公司及个人共同开发而成。由于有了Sun 的参与和支持，最新的Servlet 和JSP 规范总是能在Tomcat 中得到体现，Tomcat 5支持最新的Servlet 2.4 和JSP 2.0 规范。因为Tomcat 技术先进、性能稳定，而且免费，因而深受Java 爱好者的喜爱并得到了部分软件开发商的认可，成为目前比较流行的Web 应用服务器。

Tomcat 服务器是一个免费的开放源代码的Web 应用服务器，属于轻量级应用服务器，在中小型系统和并发访问用户不是很多的场合下被普遍使用，是开发和调试JSP 程序的首选。对于一个初学者来说，可以这样认为，当在一台机器上配置好Apache 服务器，可利用它响应HTML（标准通用标记语言下的一个应用）页面的访问请求。实际上Tomcat是Apache 服务器的扩展，但运行时它是独立运行的，所以当你运行tomcat 时，它实际上作为一个与Apache 独立的进程单独运行的。

诀窍是，当配置正确时，Apache 为HTML页面服务，而Tomcat 实际上运行JSP 页面和Servlet。另外，Tomcat和IIS等Web服务器一样，具有处理HTML页面的功能，另外它还是一个Servlet和JSP容器，独立的Servlet容器是Tomcat的默认模式。不过，Tomcat处理静态HTML的能力不如Apache服务器。目前Tomcat最新版本为9.0。

# 2.3 MVC Model 2 模式

Web 向软件开发人员提出了一些特有的挑战，最明显的就是客户机和服务器的无状态连接。这种无状态行为使得模型很难将更改通知视图。在 Web 上，为了发现对应用程序状态的修改，浏览器必须重新查询服务器。

View接受用户输入,并传递到Controller.

Controller统一进行处理命令,交由Model处理具体的业务.

经过处理Model更新后,Controller会选一个View并把Model内容传递(request,session)给它(forward).然后View进行显示.

这种模式相对MVC优势很明显，首先Model和View无需继承甚么东西，其次Model和View无需了解对方的存在，只需准备相应的接口而已；缺点是Controller层变得相对复杂了。

# 2.4 数据持久化

### 2.4.1 Mysql 数据库

MySQL是一个关系型数据库管理系统，由瑞典MySQL AB 公司开发，目前属于 Oracle 旗下产品。MySQL 是最流行的关系型数据库管理系统之一，在 WEB 应用方面，MySQL是最好的 RDBMS (Relational Database Management System，关系数据库管理系统) 应用软件。

MySQL是一种关系数据库管理系统，关系数据库将数据保存在不同的表中，而不是将所有数据放在一个大仓库内，这样就增加了速度并提高了灵活性。

MySQL所使用的 SQL 语言是用于访问数据库的最常用标准化语言。MySQL 软件采用了双授权政策，分为社区版和商业版，由于其体积小、速度快、总体拥有成本低，尤其是开放源码这一特点，一般中小型网站的开发都选择 MySQL 作为网站数据库。

由于其社区版的性能卓越，搭配 PHP 和 Apache 可组成良好的开发环境。

### 2.4.2 Mybatis

MyBatis 是一款优秀的持久层框架，它支持定制化 SQL、存储过程以及高级映射。MyBatis 避免了几乎所有的 JDBC 代码和手动设置参数以及获取结果集。MyBatis 可以使用简单的 XML 或注解来配置和映射原生信息，将接口和 Java 的 POJOs(Plain Old Java Objects,普通的 Java对象)映射成数据库中的记录。

MyBatis是支持普通 SQL查询，存储过程和高级映射的优秀持久层框架。MyBatis 消除了几乎所有的JDBC代码和参数的手工设置以及结果集的检索。MyBatis 使用简单的 XML或注解用于配置和原始映射，将接口和 Java 的POJOs（Plain Ordinary Java Objects，普通的 Java对象）映射成数据库中的记录。

每个MyBatis应用程序主要都是使用SqlSessionFactory实例的，一个SqlSessionFactory实例可以通过SqlSessionFactoryBuilder获得。SqlSessionFactoryBuilder可以从一个xml配置文件或者一个预定义的配置类的实例获得。

用xml文件构建SqlSessionFactory实例是非常简单的事情。推荐在这个配置中使用类路径资源（classpath resource)，但你可以使用任何Reader实例，包括用文件路径或file://开头的url创建的实例。MyBatis有一个实用类----Resources，它有很多方法，可以方便地从类路径及其它位置加载资源。

# 2.5 Web前端技术

### 2.5.1 HTML超文本标记语言

超级文本标记语言是标准通用标记语言下的一个应用，也是一种规范，一种标准，它通过标记符号来标记要显示的网页中的各个部分。网页文件本身是一种文本文件，通过在文本文件中添加标记符，可以告诉浏览器如何显示其中的内容（如：文字如何处理，画面如何安排，图片如何显示等）。浏览器按顺序阅读网页文件，然后根据标记符解释和显示其标记的内容，对书写出错的标记将不指出其错误，且不停止其解释执行过程，编制者只能通过显示效果来分析出错原因和出错部位。但需要注意的是，对于不同的浏览器，对同一标记符可能会有不完全相同的解释，因而可能会有不同的显示效果。

### 2.5.2 CSS层叠样式层

层叠样式表(英文全称：Cascading Style Sheets)是一种用来表现HTML（标准通用标记语言的一个应用）或XML（标准通用标记语言的一个子集）等文件样式的计算机语言。CSS不仅可以静态地修饰网页，还可以配合各种脚本语言动态地对网页各元素进行格式化。

CSS 能够对网页中元素位置的排版进行像素级精确控制，支持几乎所有的字体字号样式，拥有对网页对象和模型样式编辑的能力。

### 2.5.2 JavaScript 脚本语言

JavaScript是一种属于网络的脚本语言,已经被广泛用于Web应用开发,常用来为网页添加各式各样的动态功能,为用户提供更流畅美观的浏览效果。通常JavaScript脚本是通过嵌入在HTML中来实现自身的功能的。是一种解释性脚本语言（代码不进行预编译）。主要用来向HTML（标准通用标记语言下的一个应用）页面添加交互行为。可以直接嵌入HTML页面，但写成单独的js文件有利于结构和行为的分离。跨平台特性，在绝大多数浏览器的支持下，可以在多种平台下运行（如Windows、Linux、Mac、Android、iOS等）。

Javascript脚本语言同其他语言一样，有它自身的基本数据类型，表达式和算术运算符及程序的基本程序框架。Javascript提供了四种基本的数据类型和两种特殊数据类型用来处理数据和文字。而变量提供存放信息的地方，表达式则可以完成较复杂的信息处理。

# 2.6 运行平台

### 2.6.1 Centos Linux 服务器

CentOS 是一个基于Red Hat Linux 提供的可自由使用源代码的企业级Linux发行版本。每个版本的 CentOS都会获得十年的支持（通过安全更新方式）。新版本的 CentOS 大约每两年发行一次，而每个版本的 CentOS 会定期（大概每六个月）更新一次，以便支持新的硬件。这样，建立一个安全、低维护、稳定、高预测性、高重复性的 Linux 环境。[1] CentOS是Community Enterprise Operating System的缩写。

CentOS 是RHEL（Red Hat Enterprise Linux）源代码再编译的产物，而且在RHEL的基础上修正了不少已知的 Bug ，相对于其他 Linux 发行版，其稳定性值得信赖。

### 2.6.2 阿里云平台

阿里云创立于2009年，是全球领先的云计算及人工智能科技公司，致力于以在线公共服务的方式，提供安全、可靠的计算和数据处理能力，让计算和人工智能成为普惠科技。

阿里云服务着制造、金融、政务、交通、医疗、电信、能源等众多领域的领军企业，包括中国联通、12306、中石化、中石油、飞利浦、华大基因等大型企业客户，以及微博、知乎、锤子科技等明星互联网公司。在天猫双11全球狂欢节、12306春运购票等极富挑战的应用场景中，阿里云保持着良好的运行纪录。

阿里云在全球各地部署高效节能的绿色数据中心，利用清洁计算为万物互联的新世界提供源源不断的能源动力，目前开服的区域包括中国（华北、华东、华南、香港）、新加坡、美国（美东、美西）、欧洲、中东、澳大利亚、日本。

3 系统需求分析与设计

# 3.1需求分析与框架设计

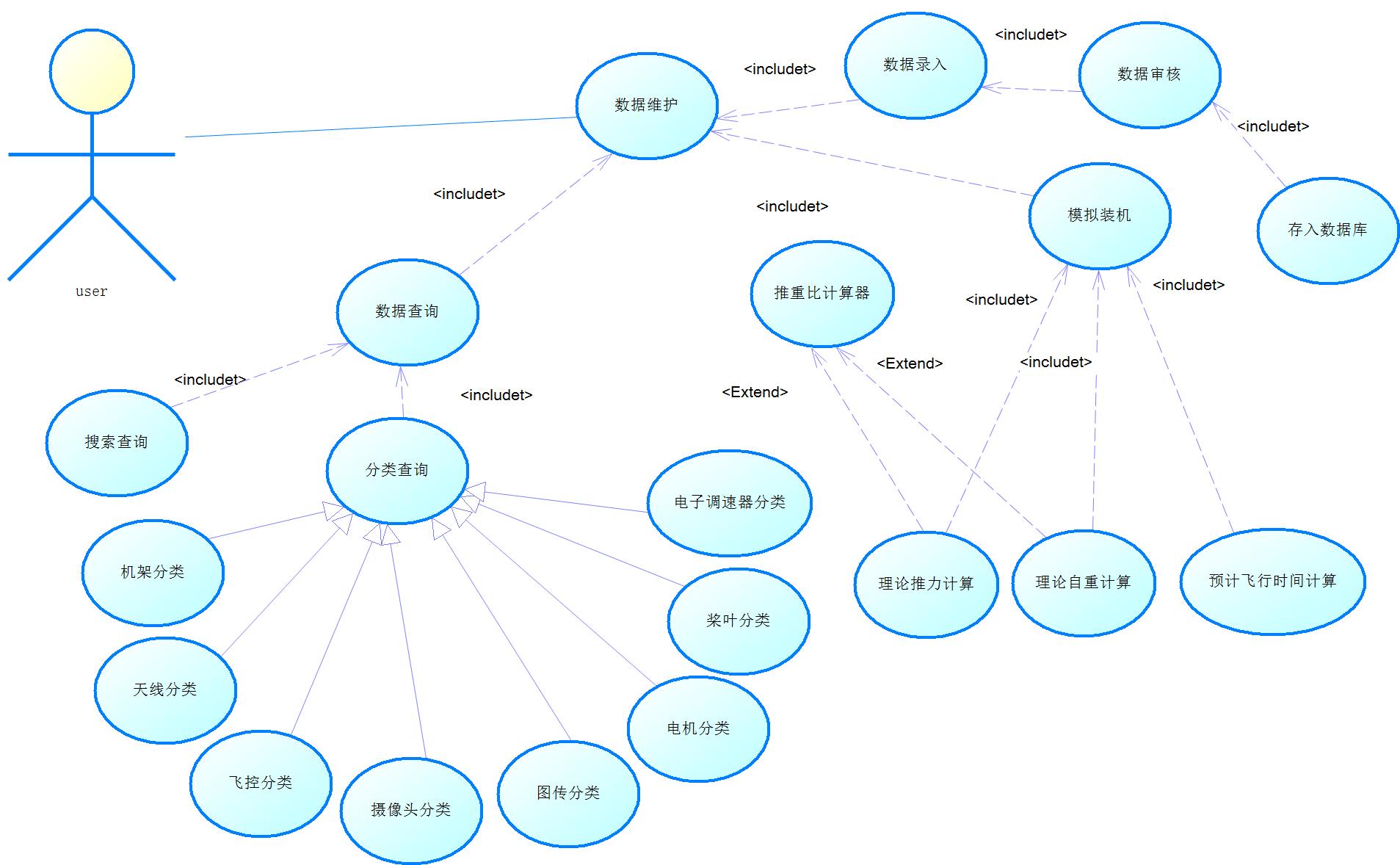
### 3.1.1总体需求分析

基于java web 及相关技术构建BS应用程序。实现在线搜索查询竞速无人机相关产品信息，在线虚拟组装并计算理论最大推力，理论飞机自重，理论飞行时间等信息。

开放数据库的数据维护及新产品信息录入开放接口，使用户参与到数据库建设中，促进形成完善的信息录入体系。用户可通过网站中的信息录入页面进行信息录入，在信息经过审核后会加入到数据库中供其他玩家使用。

# 3.1.2用户需求分析

对于无人机玩家用户系统用例图如图所示



### 在线数据查询

数据查询主要分为两种查询方式，模糊查询，即用户输入自己要查询的设备型号或设备制造商等信息进行模糊查找；以及分类查找主要包括电机、电子调速器、飞行控制器、图像传输模块、摄像头、分电板、无人机机架、电池、螺旋桨，天线等分类。玩家可利用数据库进行设备性能比较。

### 在线虚拟组装

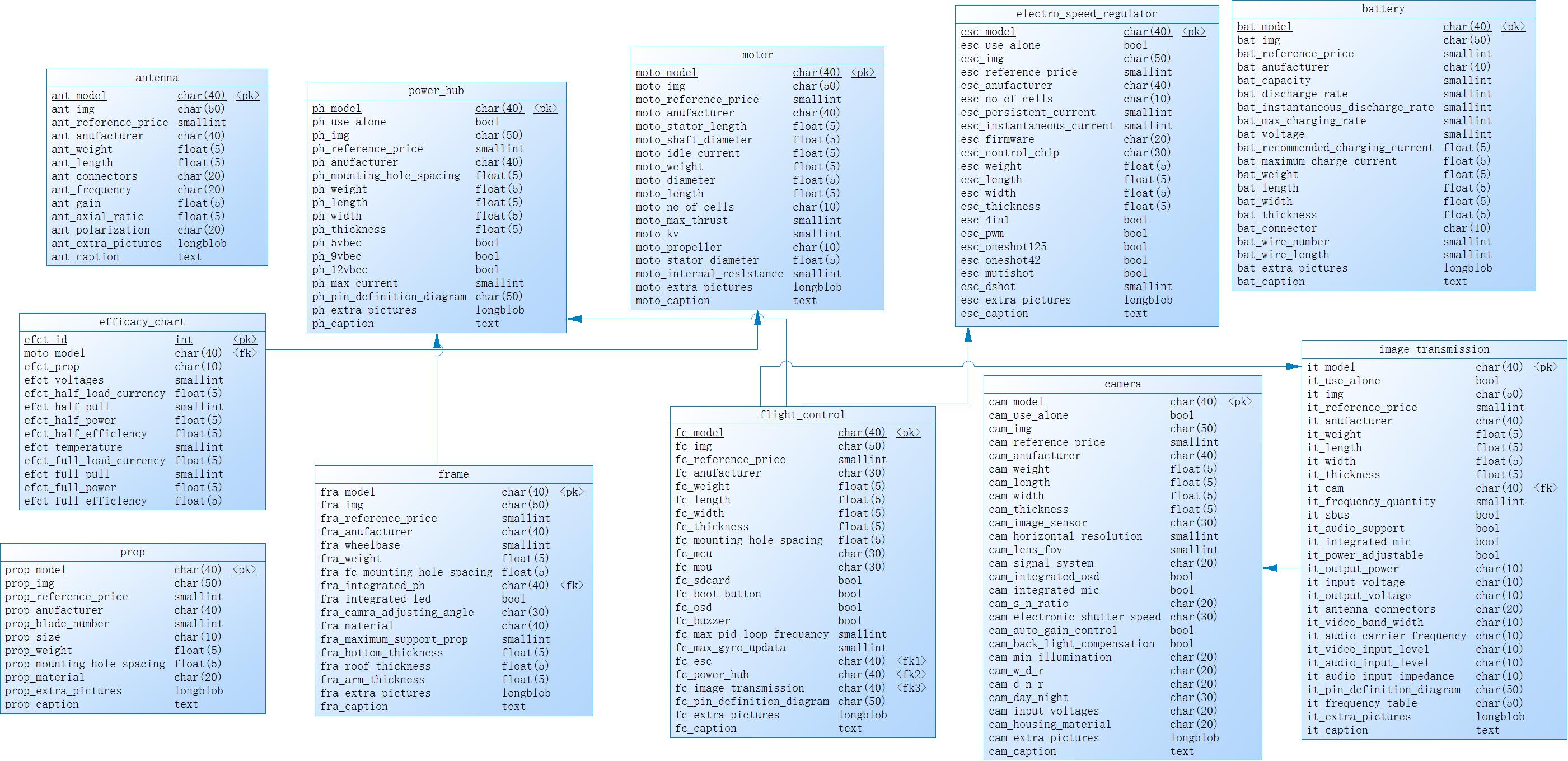
本程序提供一个可以在线虚拟组装竞速无人机的工具，玩家可以在数据库中检索自己需要的设备并利用虚拟装机工具进行虚拟装机。本工具可帮助玩家进行设备选择，主动帮助玩家避免设备之间的不适配，以及由搭配不均衡导致的某个模块的性能过剩，同时可主动帮助玩家进行用电分析并优化供电方案。

在进行虚拟组装后本程序还可以跟所选设备的参数计算出飞机的重量，最大推力，推重比，悬停飞行时间，航线飞行时间，以及暴力竞速飞行时间。根据此功能玩家可以在进行飞机组装前了解飞机的整体性能，并最大限度地优化自己的飞机。

### 数据录入

秉承着无人机开源互助的大环境同时也为了保证数据库数据的专业、精准以及更新的及时，我们设计了开放式的数据维护平台，对玩家开放数据库的数据录入接口和数据修改接口，只要玩家按照我们提供的数据格式和单位提供正确的数据在经过审核无误后都将被程序采用。

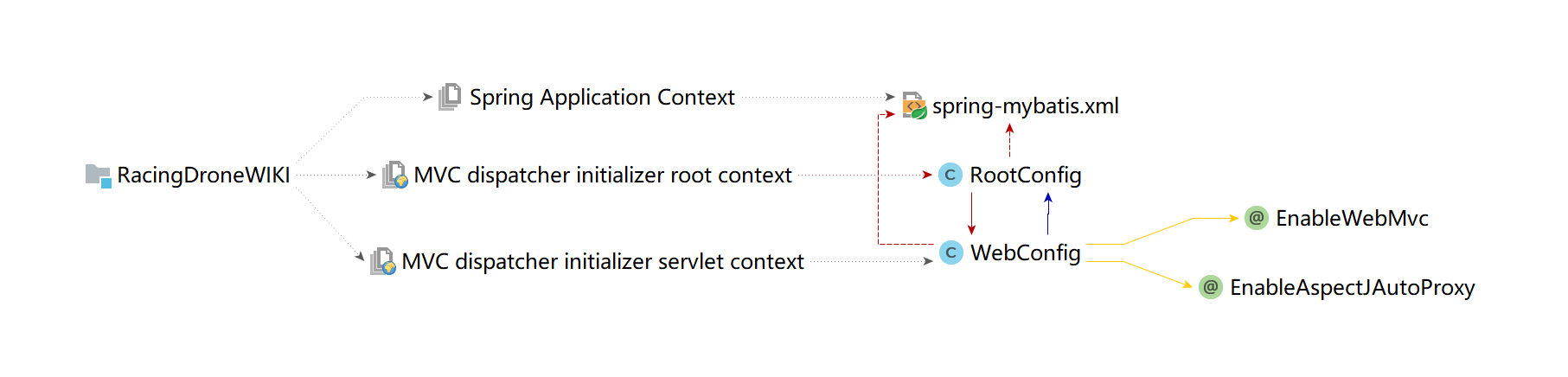
# 3.2 竞速无人机在线资料库数据库设计

根据需求分析以及个模块的功能对数据库进行详细，设计数据库设计图如下。

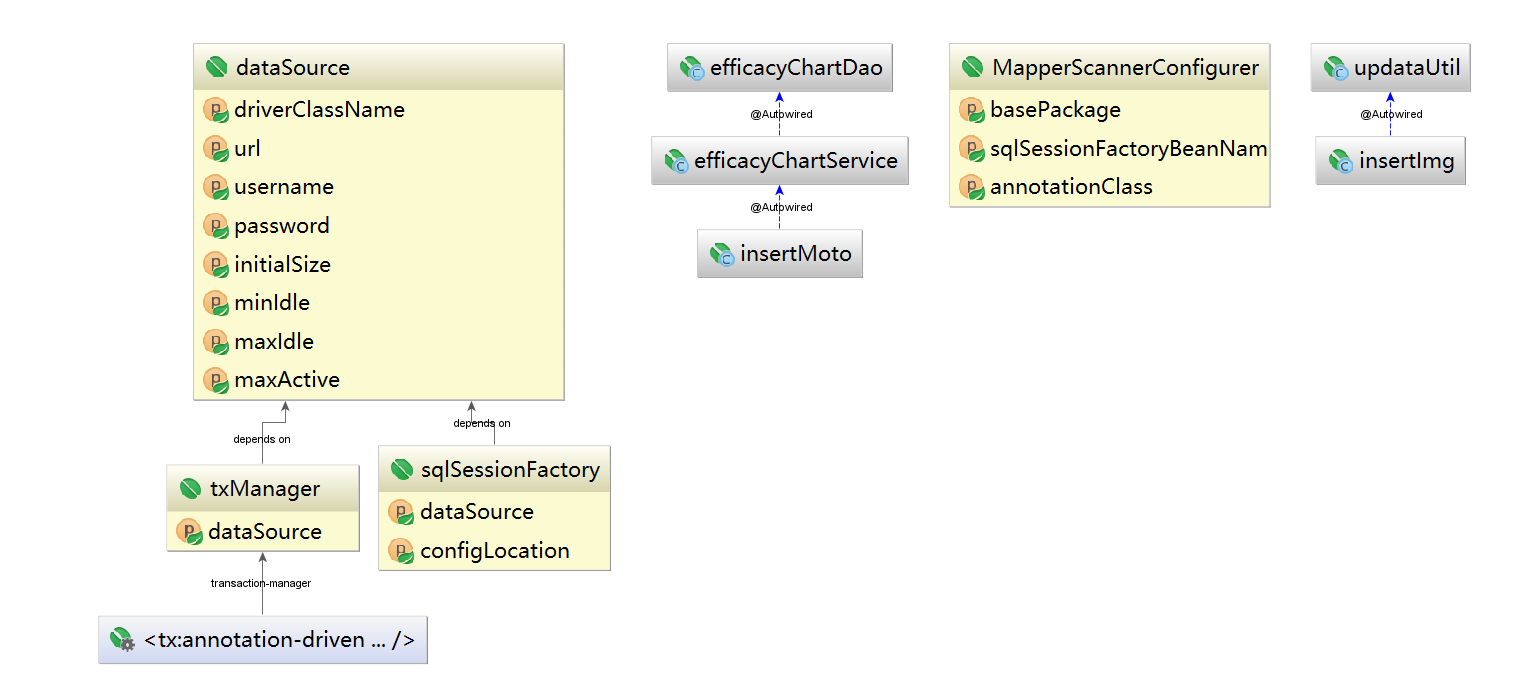
# 3.3 框架设计与实现’

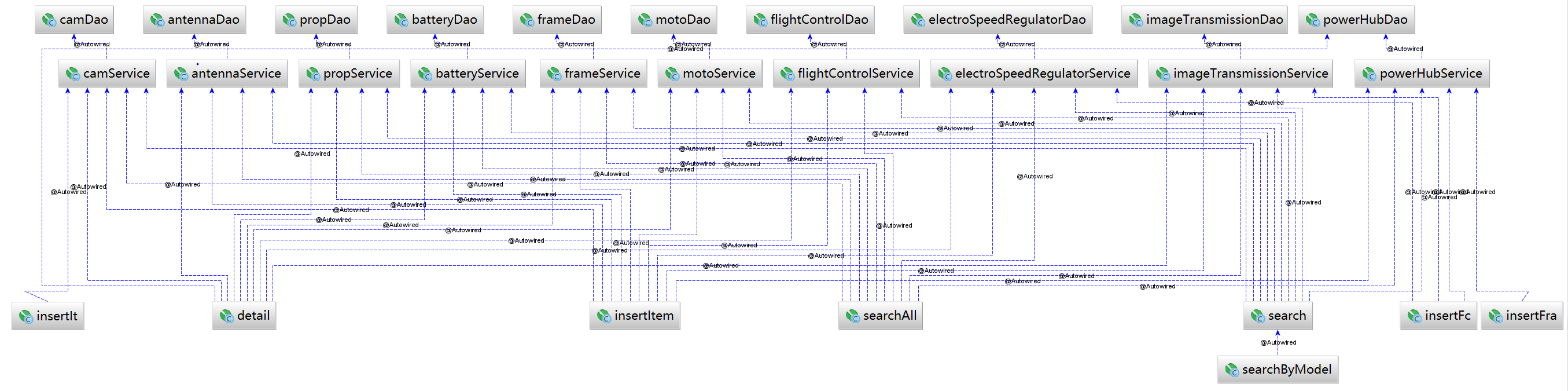
本项目使用SSM（Spring+SpringMVC+MyBatis）框架实现。其中spring是一个轻量级的控制反转（IoC）和面向切面（AOP）的容器框架。SpringMVC分离了控制器、模型对象、分派器以及处理程序对象的角色，这种分离让它们更容易进行定制。MyBatis是一个支持普通SQL查询，存储过程和高级映射的优秀持久层框架。

### Spring SpringMVC Mybatis整合结构



### Bean配置及引用关系





# 3.4界面设计

### 主界面



### 列表展示界面

### 详细信息界面

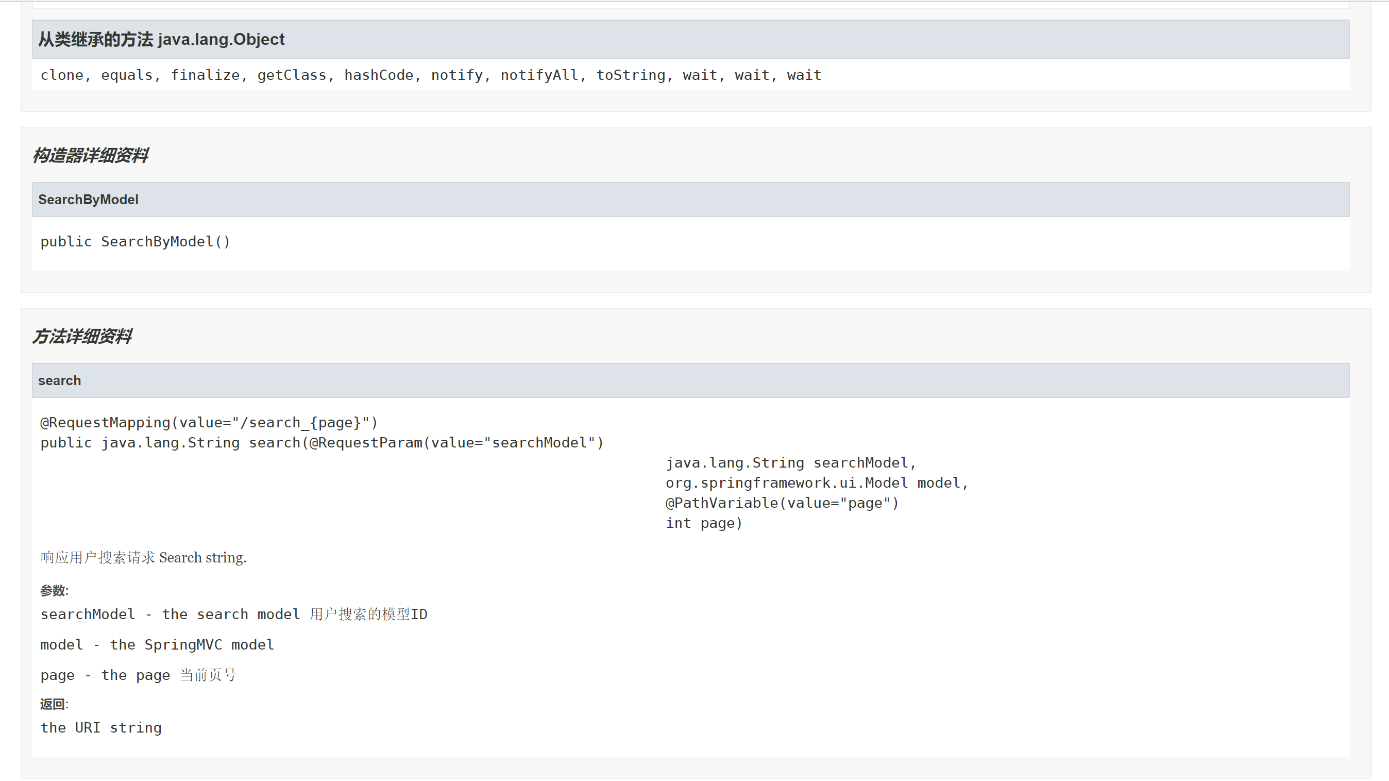
### 

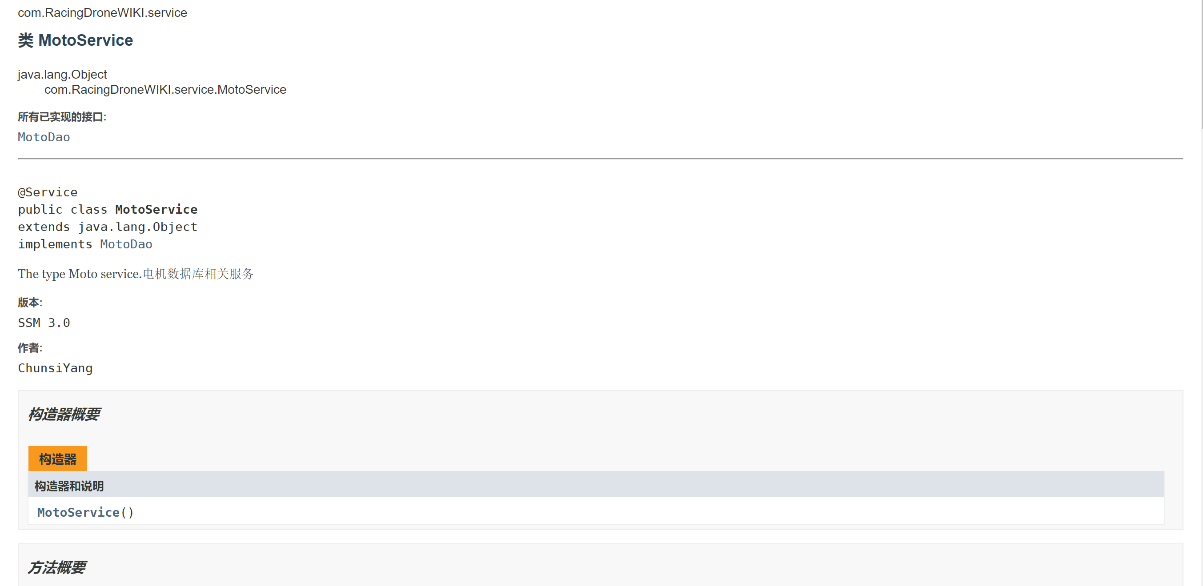
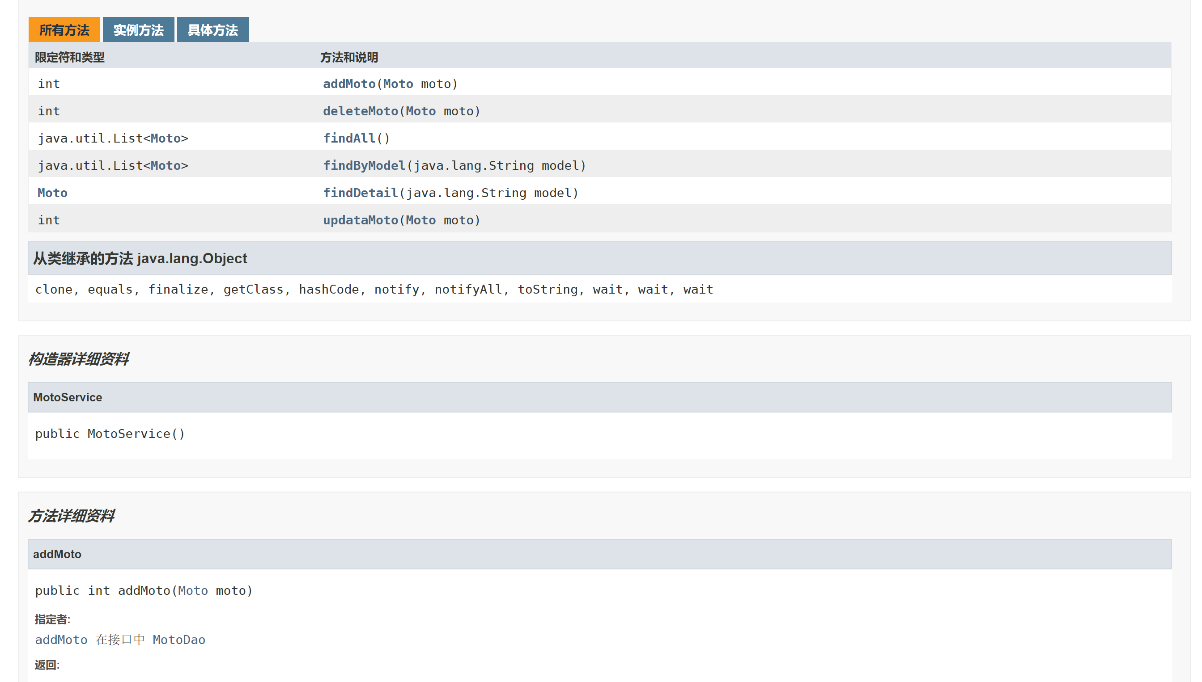
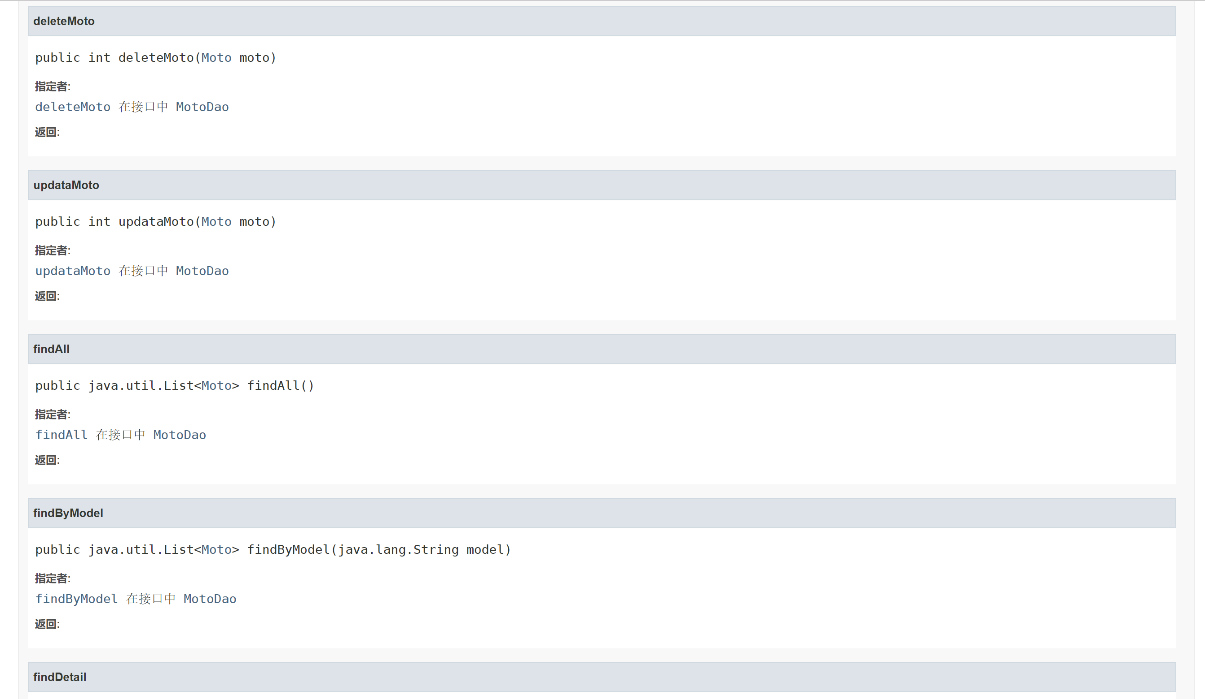
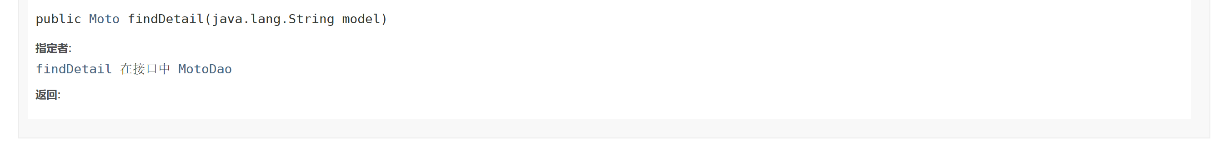
### 数据录入选择界面

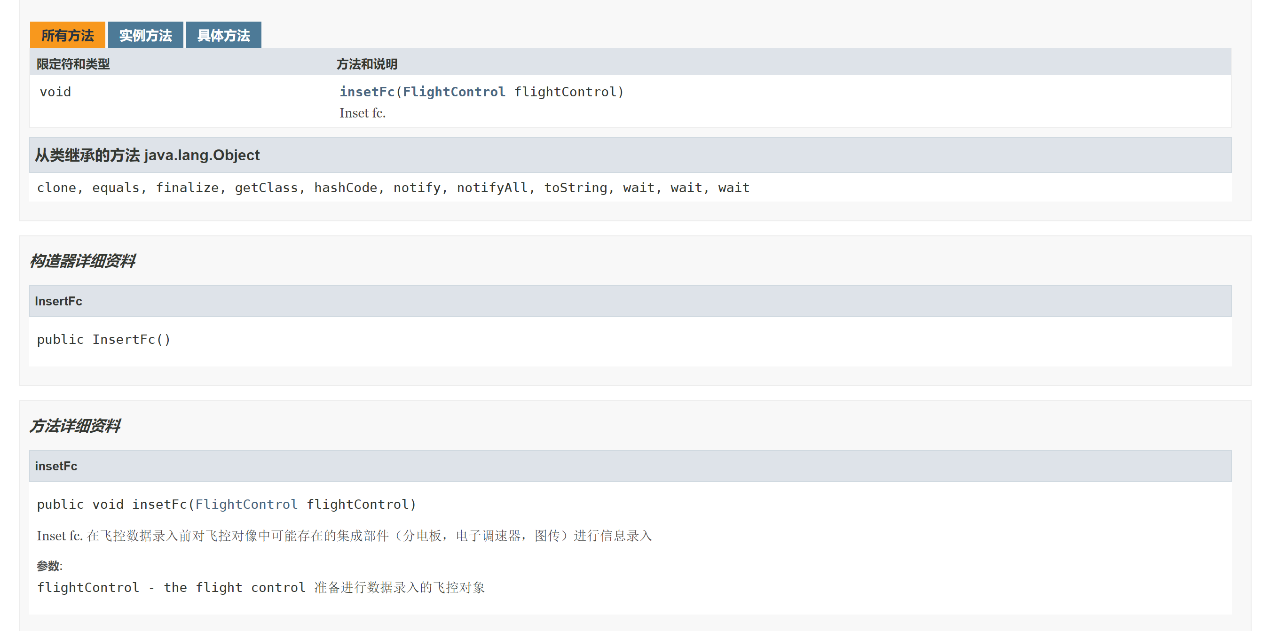
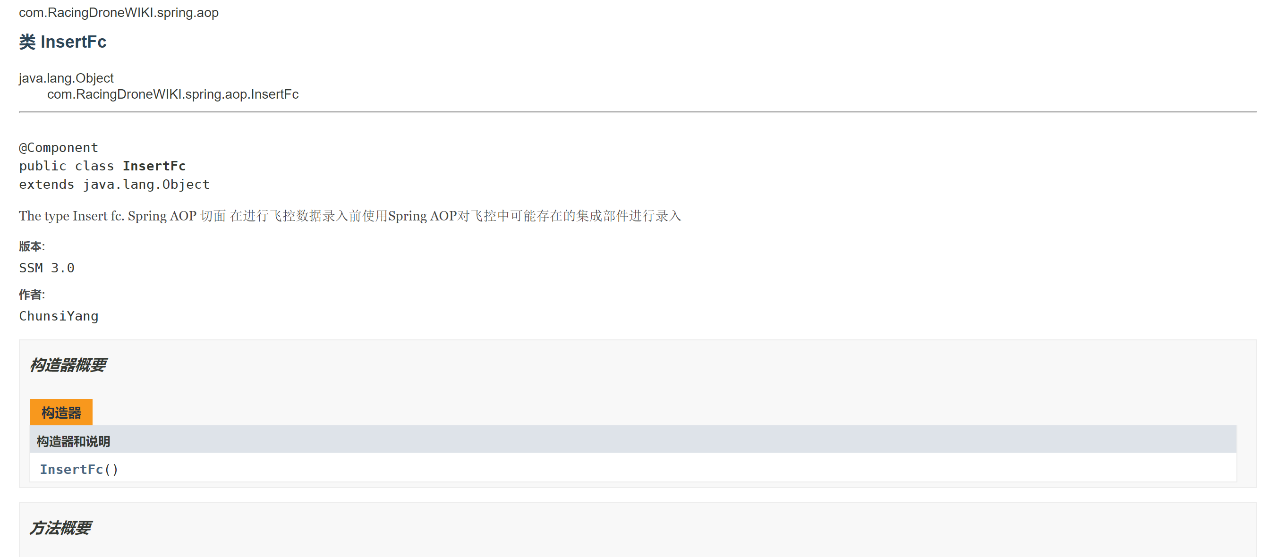
### 数据录入界面



# 核心模块Java Doc







4.结论

本程序基于SSM框架实现，为广大竞速无人机爱好者提供了一个开放但并不无序的信息资料分享平台。基于竞速无人机圈内的开源特性采用了完全开放的数据管理模式。本程序目前还不能支持高效的数据自动录入，只能通过手动进行录入，在未来会添加自动化数据录入。且会添加更多基于数据的模拟信息和应用。

5．核心代码

# 主页面

<!DOCTYPE **html**>  
<**html lang="en" xmlns:jsp="http://java.sun.com/jsf/facelets"**>  
<**head**>  
 <**meta charset="UTF-8"**>  
 <**title**>竞速无人机资料库</**title**>  
 <**link type="text/css" rel="stylesheet" href="mainHorizontal.css"**>  
</**head**>  
<**body**>  
<**section id="main"** >  
 <**a id="dataInput" href="itemInsert/dataInput"**>数据录入</**a**>  
 <**h1**>竞速无人机资料库</**h1**>  
 <**form id="searchBar" method="get" action="/search\_1"**>  
 <**input id="search" type="text" name="searchModel" placeholder=" 输入设备型号搜索..."**>  
 <**input id="searchButton" type="submit" value="搜索"**>  
 </**form**>  
 <**nav**>  
 <**button class="item" onclick="javascript**:**window**.**location**.**href**=**'/moto\_1'**;**"** >  
 <**a href="/moto\_1"** >  
 <**img src="img/moto.png" class="itemImg"**/>  
 <**p**>电机</**p**>  
 </**a**>  
 </**button**>  
 <**button class="item" onclick="javascript**:**window**.**location**.**href**=**'/electroSpeedRegulator\_1'**;**"**>  
 <**a href="/electroSpeedRegulator\_1"** >  
 <**img src="img/esc.jpg" class="itemImg"**/>  
 <**p**>电调</**p**>  
 </**a**>  
 </**button**>  
 <**button class="item" onclick="javascript**:**window**.**location**.**href**=**'/frame\_1'**;**"**>  
 <**a href="/frame\_1"** >  
 <**img src="img/fra.jpg" class="itemImg"**/>  
 <**p**>机架</**p**>  
 </**a**>  
 </**button**>  
 <**button class="item" onclick="javascript**:**window**.**location**.**href**=**'/imageTransmission\_1'**;**"**>  
 <**a href="/imageTransmission\_1"** >  
 <**img src="img/it.jpg" class="itemImg"**/>  
 <**p**>图传</**p**>  
 </**a**>  
 </**button**>  
 <**button class="item" onclick="javascript**:**window**.**location**.**href**=**'/cam\_1'**;**"**>  
 <**a href="/cam\_1"** >  
 <**img src="img/cam.jpg" class="itemImg"**/>  
 <**p**>摄像头</**p**>  
 </**a**>  
 </**button**>  
 <**br**>  
 <**button class="item" onclick="javascript**:**window**.**location**.**href**=**'/flightControl\_1'**;**"**>  
 <**a href="/flightControl\_1"** >  
 <**img src="img/fc.jpg" class="itemImg"**/>  
 <**p**>飞控</**p**>  
 </**a**>  
 </**button**>  
 <**button class="item" onclick="javascript**:**window**.**location**.**href**=**'/antenna\_1'**;**"**>  
 <**a href="/antenna\_1"** >  
 <**img src="img/ant.jpg" class="itemImg"**/>  
 <**p**>天线</**p**>  
 </**a**>  
 </**button**>  
 <**button class="item" onclick="javascript**:**window**.**location**.**href**=**'/powerHub\_1'**;**"**>  
 <**a href="/powerHub\_1"** >  
 <**img src="img/ph.jpg" class="itemImg"**/>  
 <**p**>分电板</**p**>  
 </**a**>  
 </**button**>  
 <**button class="item" onclick="javascript**:**window**.**location**.**href**=**'/prop\_1'**;**"**>  
 <**a href="/prop\_1"** >  
 <**img src="img/prop.png" class="itemImg"**/>  
 <**p**>桨叶</**p**>  
 </**a**>  
 </**button**>  
 <**button class="item" onclick="javascript**:**window**.**location**.**href**=**'/battery\_1'**;**"**>  
 <**a href="/battery\_1"** >  
 <**img src="img/bat.jpg" class="itemImg"**/>  
 <**p**>电池</**p**>  
 </**a**>  
 </**button**>  
 </**nav**>  
 <%@**include file**="**foot.html**"%>  
</**section**>  
</**body**>  
</**html**>

# 数据搜索

**package** com.RacingDroneWIKI.controller;  
  
**import** com.RacingDroneWIKI.pojo.\*;  
**import** com.RacingDroneWIKI.service.Search;  
**import** org.springframework.beans.factory.annotation.Autowired;  
**import** org.springframework.stereotype.Controller;  
**import** org.springframework.ui.Model;  
**import** org.springframework.web.bind.annotation.PathVariable;  
**import** org.springframework.web.bind.annotation.RequestMapping;  
**import** org.springframework.web.bind.annotation.RequestParam;  
  
**import** java.util.Iterator;  
**import** java.util.LinkedList;  
**import** java.util.List;  
  
*/\*\*  
 \* 响应用户搜索请求的Controller  
 \* 对所有类型的设备进行搜素并展示给用户  
 \** ***@author*** *ChunsiYang  
 \** ***@version*** *SSM 3.0  
 \*/*@Controller  
**public class** SearchByModel {  
 **private final int** PAGENUMBER = 10;  
 @Autowired  
 **private** Search search;  
  
 */\*\*  
 \* 响应用户搜索请求  
 \* Search string.  
 \*  
 \** ***@param*** *searchModel the search model 用户搜索的模型ID  
 \** ***@param*** *model the SpringMVC model  
 \** ***@param*** *page the page 当前页号  
 \** ***@return*** *the URI string  
 \*/* @RequestMapping(value = **"/search\_{page}"**)  
 **public** String search(@RequestParam(**"searchModel"**) String searchModel, Model model, @PathVariable(value = **"page"**) **int** page) {  
 List<Moto> motoList = **new** LinkedList<>();  
 List<ElectroSpeedRegulator> electroSpeedRegulatorList = **new** LinkedList<>();  
 List<Frame> frameList = **new** LinkedList<>();  
 List<ImageTransmission> imageTransmissionList = **new** LinkedList<>();  
 List<Cam> camList = **new** LinkedList<>();  
 List<FlightControl> flightControlList = **new** LinkedList<>();  
 List<Antenna> antennaList = **new** LinkedList<>();  
 List<PowerHub> powerHubList = **new** LinkedList<>();  
 List<Prop> propList = **new** LinkedList<>();  
 List<Battery> batteryList = **new** LinkedList<>();  
 search.searchModel(searchModel, motoList, electroSpeedRegulatorList, frameList,  
 imageTransmissionList, camList, flightControlList, antennaList,  
 powerHubList, propList, batteryList);  
 **int** pages;  
 **int** first;  
 **int** last;  
 **int** size = motoList.size() + electroSpeedRegulatorList.size()  
 + frameList.size() + imageTransmissionList.size() +  
 camList.size() + flightControlList.size() + antennaList.size()  
 + powerHubList.size() + propList.size() + batteryList.size();  
  
 pages = size % PAGENUMBER != 0 ? size / PAGENUMBER + 1 : size / PAGENUMBER;  
 first = (page - 1) \* PAGENUMBER;  
 last = size > (first + PAGENUMBER) ? first + PAGENUMBER : size;  
 List<Item> list = **new** LinkedList();  
 list.addAll(motoList);  
 list.addAll(electroSpeedRegulatorList);  
 list.addAll(frameList);  
 list.addAll(imageTransmissionList);  
 list.addAll(camList);  
 list.addAll(flightControlList);  
 list.addAll(antennaList);  
 list.addAll(powerHubList);  
 list.addAll(propList);  
 list.addAll(batteryList);  
 list = list.subList(first, last);  
 motoList.clear();  
 electroSpeedRegulatorList.clear();  
 frameList.clear();  
 imageTransmissionList.clear();  
 camList.clear();  
 flightControlList.clear();  
 antennaList.clear();  
 powerHubList.clear();  
 propList.clear();  
 batteryList.clear();  
 Iterator<Item> itemIterator = list.iterator();  
 **while** (itemIterator.hasNext()) {  
 Item item = itemIterator.next();  
 **if** (item **instanceof** Moto) {  
 motoList.add((Moto) item);  
 }  
 **if** (item **instanceof** ElectroSpeedRegulator) {  
 electroSpeedRegulatorList.add((ElectroSpeedRegulator) item);  
 }  
 **if** (item **instanceof** Frame) {  
 frameList.add((Frame) item);  
 }  
 **if** (item **instanceof** ImageTransmission) {  
 imageTransmissionList.add((ImageTransmission) item);  
 }  
 **if** (item **instanceof** Cam) {  
 camList.add((Cam) item);  
 }  
 **if** (item **instanceof** FlightControl) {  
 flightControlList.add((FlightControl) item);  
 }  
 **if** (item **instanceof** Antenna) {  
 antennaList.add((Antenna) item);  
 }  
 **if** (item **instanceof** PowerHub) {  
 powerHubList.add((PowerHub) item);  
 }  
 **if** (item **instanceof** Prop) {  
 propList.add((Prop) item);  
 }  
 **if** (item **instanceof** Battery) {  
 batteryList.add((Battery) item);  
 }  
 }  
 model.addAttribute(**"uri"**, **"/search\_"**);  
 model.addAttribute(**"pages"**, pages);  
 model.addAttribute(**"page"**, page);  
 model.addAttribute(**"searchModel"**, **"?searchModel="** + searchModel);  
 model.addAttribute(**"motoList"**, motoList);  
 model.addAttribute(**"electroSpeedRegulatorList"**, electroSpeedRegulatorList);  
 model.addAttribute(**"frameList"**, frameList);  
 model.addAttribute(**"imageTransmissionList"**, imageTransmissionList);  
 model.addAttribute(**"camList"**, camList);  
 model.addAttribute(**"flightControlList"**, flightControlList);  
 model.addAttribute(**"antennaList"**, antennaList);  
 model.addAttribute(**"powerHubList"**, powerHubList);  
 model.addAttribute(**"propList"**, propList);  
 model.addAttribute(**"batteryList"**, batteryList);  
 **return "resultSet"**;  
 }  
}

# 数据录入

**package** com.RacingDroneWIKI.controller;  
  
**import** com.RacingDroneWIKI.pojo.\*;  
**import** com.RacingDroneWIKI.service.\*;  
**import** org.springframework.beans.factory.annotation.Autowired;  
**import** org.springframework.stereotype.Controller;  
**import** org.springframework.transaction.annotation.Propagation;  
**import** org.springframework.transaction.annotation.Transactional;  
**import** org.springframework.web.bind.annotation.RequestMapping;  
**import** org.springframework.web.bind.annotation.RequestMethod;  
**import** org.springframework.web.bind.annotation.RequestParam;  
**import** org.springframework.web.multipart.MultipartFile;  
  
**import** javax.servlet.http.HttpServletRequest;  
**import** javax.servlet.http.HttpSession;  
**import** java.io.IOException;  
  
*/\*\*  
 \* 响应数据录入请求的Controller  
 \* 对用户的数据录入请求进行整理并启用服务添加至数据库  
 \*  
 \** ***@author*** *ChunsiYang  
 \** ***@version*** *SSM 3.0  
 \*/*@Controller  
@RequestMapping(value = **"/itemInsert/"**)  
**public class** InsertItem {  
 @Autowired  
 **private** AntennaService **antennaService**;  
 @Autowired  
 **private** BatteryService **batteryService**;  
 @Autowired  
 **private** PropService **propService**;  
 @Autowired  
 **private** CamService **camService**;  
 @Autowired  
 **private** PowerHubService **powerHubService**;  
 @Autowired  
 **private** ElectroSpeedRegulatorService **electroSpeedRegulatorService**;  
 @Autowired  
 **private** MotoService **motoService**;  
 @Autowired  
 **private** FrameService **frameService**;  
 @Autowired  
 **private** FlightControlService **flightControlService**;  
 @Autowired  
 **private** ImageTransmissionService **imageTransmissionService**;  
  
 */\*\*  
 \* 数据录入模型选择界面  
 \* Data input string.  
 \*  
 \** ***@param session*** *the HTTP session  
 \** ***@return*** *the URI string  
 \*/* @RequestMapping(**"dataInput"**)  
 **public** String dataInput(HttpSession session) {  
 session.invalidate();  
 **return "itemInsert/dataInput"**;  
 }  
  
 */\*\*  
 \* 响应天线录入请求  
 \* Insert ant string.  
 \*  
 \** ***@param antenna*** *the antenna SpringMVC绑定的前台天线数据  
 \** ***@param request*** *the HTTP servlet request  
 \** ***@param mainImg*** *the main img 主图片  
 \** ***@param files*** *the files 辅助图片  
 \** ***@return*** *the URI string  
 \** ***@throws*** *IOException the io exception  
 \*/* @RequestMapping(value = **"InsertAnt"**, method = RequestMethod.***POST***)  
 **public** String insertAnt(Antenna antenna, HttpServletRequest request,  
 @RequestParam(**"mainImg"**) MultipartFile mainImg,  
 @RequestParam(**"files"**) MultipartFile[] files) **throws** IOException {  
 **antennaService**.addAntenna(antenna);  
 **return "itemInsert/InsertSuccessed"**;  
 }  
  
 */\*\*  
 \* 响应电池录入请求  
 \* Insert bat string.  
 \*  
 \** ***@param battery*** *the battery SpringMVC绑定的前台电池数据  
 \** ***@param request*** *the HTTP servlet request  
 \** ***@param mainImg*** *the main img 主图片  
 \** ***@param files*** *the files 辅助图片  
 \** ***@return*** *the URI string  
 \** ***@throws*** *IOException the io exception  
 \*/* @RequestMapping(value = **"InsertBat"**, method = RequestMethod.***POST***)  
 **public** String insertBat(Battery battery, HttpServletRequest request,  
 @RequestParam(**"mainImg"**) MultipartFile mainImg,  
 @RequestParam(**"files"**) MultipartFile[] files) **throws** IOException {  
 **batteryService**.addBattery(battery);  
 **return "itemInsert/InsertSuccessed"**;  
 }  
  
 */\*\*  
 \* 响应桨叶录入请求  
 \* Insert prop string.  
 \*  
 \** ***@param prop*** *the prop SpringMVC绑定的前台桨叶数据  
 \** ***@param request*** *the HTTP servlet request  
 \** ***@param mainImg*** *the main img 主图片  
 \** ***@param files*** *the files 辅助图片  
 \** ***@return*** *the URI string  
 \** ***@throws*** *IOException the io exception  
 \*/* @RequestMapping(value = **"InsertProp"**, method = RequestMethod.***POST***)  
 **public** String insertProp(Prop prop, HttpServletRequest request,  
 @RequestParam(**"mainImg"**) MultipartFile mainImg,  
 @RequestParam(**"files"**) MultipartFile[] files) **throws** IOException {  
 **propService**.addProp(prop);  
 **return "itemInsert/InsertSuccessed"**;  
 }  
  
 */\*\*  
 \* 响应摄像头录入请求  
 \* Insert cam string.  
 \*  
 \** ***@param cam*** *the cam SpringMVC绑定的前台摄像头数据  
 \** ***@param request*** *the HTTP servlet request  
 \** ***@param mainImg*** *the main img 诸如片  
 \** ***@param files*** *the files 辅助图片  
 \** ***@return*** *the URI string  
 \** ***@throws*** *IOException the io exception  
 \*/* @RequestMapping(value = **"InsertCam"**, method = RequestMethod.***POST***)  
 **public** String insertCam(Cam cam, HttpServletRequest request,  
 @RequestParam(**"mainImg"**) MultipartFile mainImg,  
 @RequestParam(**"files"**) MultipartFile[] files) **throws** IOException {  
 HttpSession session = request.getSession();  
 ImageTransmission imageTransmission = (ImageTransmission) session.getAttribute(**"imageTransmission"**);  
 **if** ((imageTransmission != **null**)) {  
 imageTransmission.setCam(cam);  
 session.setAttribute(**"imageTransmission"**, imageTransmission);  
 **return "itemInsert/ItIntergrat"**;  
 } **else** {  
 cam.setUseAlone(**true**);  
 **camService**.addCam(cam);  
 **return "itemInsert/InsertSuccessed"**;  
 }  
 }  
  
 */\*\*  
 \* 响应分电板录入请求  
 \* Insert ph string.  
 \*  
 \** ***@param powerHub*** *the power hub SpringMVC绑定的前台分电板数据  
 \** ***@param request*** *the HTTP servlet request  
 \** ***@param mainImg*** *the main img 主图片  
 \** ***@param files*** *the files 辅助图片  
 \** ***@param pinDef*** *the pin def 引脚定义图  
 \** ***@return*** *the URI string  
 \** ***@throws*** *IOException the io exception  
 \*/* @RequestMapping(value = **"InsertPh"**, method = RequestMethod.***POST***)  
 **public** String insertPH(PowerHub powerHub, HttpServletRequest request,  
 @RequestParam(**"mainImg"**) MultipartFile mainImg,  
 @RequestParam(**"files"**) MultipartFile[] files,  
 @RequestParam(**"pinDefinitionDiagramUrl"**) MultipartFile pinDef) **throws** IOException {  
 HttpSession session = request.getSession();  
 Frame frame = (Frame) session.getAttribute(**"frame"**);  
 FlightControl flightControl = (FlightControl) session.getAttribute(**"flightControl"**);  
 **if** ((frame != **null**)) {  
 frame.setIntegratedPh(powerHub);  
 session.setAttribute(**"frame"**, frame);  
 **return "itemInsert/FraIntergrat"**;  
 }  
 **if** ((flightControl != **null**)) {  
 flightControl.setPowerHub(powerHub);  
 session.setAttribute(**"flightControl"**, flightControl);  
 **return "itemInsert/FcIntergrat"**;  
 }  
 powerHub.setUseAlone(**true**);  
 **powerHubService**.addPowerHub(powerHub);  
 **return "itemInsert/InsertSuccessed"**;  
 }  
  
 */\*\*  
 \* 响应电子调速器录入请求  
 \* Insert esc string.  
 \*  
 \** ***@param electroSpeedRegulator*** *the electro speed regulator SpringMVC绑定的前台电子调速器数据  
 \** ***@param request*** *the HTTP servlet request  
 \** ***@param mainImg*** *the main img 主图片  
 \** ***@param files*** *the files 辅助图片  
 \** ***@return*** *the URI string  
 \** ***@throws*** *IOException the io exception  
 \*/* @RequestMapping(value = **"InsertEsc"**, method = RequestMethod.***POST***)  
 **public** String insertEsc(ElectroSpeedRegulator electroSpeedRegulator, HttpServletRequest request,  
 @RequestParam(**"mainImg"**) MultipartFile mainImg,  
 @RequestParam(**"files"**) MultipartFile[] files) **throws** IOException {  
 HttpSession session = request.getSession();  
 FlightControl flightControl = (FlightControl) session.getAttribute(**"flightControl"**);  
 **if** ((flightControl != **null**)) {  
 flightControl.setEsc(electroSpeedRegulator);  
 session.setAttribute(**"flightControl"**, flightControl);  
 **return "itemInsert/FcIntergrat"**;  
 }  
 electroSpeedRegulator.setUseAlone(**true**);  
 **electroSpeedRegulatorService**.addElectroSpeedRegulato(electroSpeedRegulator);  
 **return "itemInsert/InsertSuccessed"**;  
 }  
  
 */\*\*  
 \* 响应电机录入请求  
 \* Insert moto string.  
 \*  
 \** ***@param moto*** *the moto SpringMVC绑定的前台电机数据  
 \** ***@param request*** *the HTTP servlet request  
 \** ***@param mainImg*** *the main img 主图片  
 \** ***@param files*** *the files 辅助图片  
 \** ***@return*** *the URI string  
 \** ***@throws*** *IOException the io exception  
 \*/* @RequestMapping(value = **"InsertMoto"**, method = RequestMethod.***POST***)  
 **public** String insertMoto(Moto moto, HttpServletRequest request,  
 @RequestParam(**"mainImg"**) MultipartFile mainImg,  
 @RequestParam(**"files"**) MultipartFile[] files) **throws** IOException {  
 **motoService**.addMoto(moto);  
 **return "itemInsert/InsertSuccessed"**;  
 }  
  
 */\*\*  
 \* 响应机架中集成部件录入请求  
 \* Insert fra string.  
 \*  
 \** ***@param frame*** *the frame SpringMVC绑定的前台机架数据  
 \** ***@param request*** *the HTTP servlet request  
 \** ***@param mainImg*** *the main img 主图片  
 \** ***@param files*** *the files 辅助图片  
 \** ***@return*** *the URI string  
 \** ***@throws*** *IOException the io exception  
 \*/* @RequestMapping(value = **"InsertFra"**, method = RequestMethod.***POST***)  
 **public** String insertFra(Frame frame, HttpServletRequest request,  
 @RequestParam(**"mainImg"**) MultipartFile mainImg,  
 @RequestParam(**"files"**) MultipartFile[] files) **throws** IOException {  
 HttpSession session = request.getSession();  
 session.setAttribute(**"frame"**, frame);  
 **return "itemInsert/FraIntergrat"**;  
 }  
  
 */\*\*  
 \* 响应机架录入请求  
 \* Insert fra string.  
 \*  
 \** ***@param session*** *the HTTP servlet session  
 \** ***@return*** *the URI string  
 \*/* @Transactional(propagation = Propagation.***REQUIRED***)  
 @RequestMapping(value = **"InsertFra"**, method = RequestMethod.***GET***)  
 **public** String insertFra(HttpSession session) {  
 Frame frame = (Frame) session.getAttribute(**"frame"**);  
 **frameService**.addFrame(frame);  
 session.invalidate();  
 **return "itemInsert/InsertSuccessed"**;  
 }  
  
 */\*\*  
 \* 响应飞控中集成部件录入请求  
 \* Inset fc string.  
 \*  
 \** ***@param flightControl*** *the flight control SpringMVC绑定的前台飞控数据  
 \** ***@param request*** *the HTTP servlet request  
 \** ***@param mainImg*** *the main img 主图片  
 \** ***@param files*** *the files 辅助图片  
 \** ***@param pinDef*** *the pin def 引脚定义图  
 \** ***@return*** *the URI string  
 \** ***@throws*** *IOException the io exception  
 \*/* @RequestMapping(value = **"InsertFC"**, method = RequestMethod.***POST***)  
 **public** String insetFc(FlightControl flightControl, HttpServletRequest request,  
 @RequestParam(**"mainImg"**) MultipartFile mainImg,  
 @RequestParam(**"files"**) MultipartFile[] files,  
 @RequestParam(**"pinDefinitionDiagramUrl"**) MultipartFile pinDef) **throws** IOException {  
 HttpSession session = request.getSession();  
 session.setAttribute(**"flightControl"**, flightControl);  
 **return "itemInsert/FcIntergrat"**;  
 }  
  
 */\*\*  
 \* 响应飞控录入秦桂青  
 \* Insert fc string.  
 \*  
 \** ***@param session*** *the HTTP servlet session  
 \** ***@return*** *the URI string  
 \*/* @Transactional(propagation = Propagation.***REQUIRED***)  
 @RequestMapping(value = **"InsertFC"**, method = RequestMethod.***GET***)  
 **public** String insertFc(HttpSession session) {  
 FlightControl flightControl = (FlightControl) session.getAttribute(**"flightControl"**);  
 flightControlService.addFlightControl(flightControl);  
 session.invalidate();  
 **return "itemInsert/InsertSuccessed"**;  
 }  
  
 */\*\*  
 \* 响应图传中集成部件的录入请求  
 \* Inset it string.  
 \*  
 \** ***@param*** *imageTransmission the image transmission SpringMVC绑定的前台图传数据  
 \** ***@param*** *request the HTTP servlet request  
 \** ***@param*** *mainImg the main img 主图片  
 \** ***@param*** *files the files 辅助图片  
 \** ***@param*** *pinDef the pin def 引脚定义图  
 \** ***@param*** *fre the fre 频率表  
 \** ***@return*** *the URI string  
 \** ***@throws*** *IOException the io exception  
 \*/* @RequestMapping(value = **"InsertIt"**, method = RequestMethod.POST)  
 **public** String insetIt(ImageTransmission imageTransmission, HttpServletRequest request,  
 @RequestParam(**"mainImg"**) MultipartFile mainImg,  
 @RequestParam(**"files"**) MultipartFile[] files,  
 @RequestParam(**"pinDefinitionDiagramUrl"**) MultipartFile pinDef,  
 @RequestParam(**"frequencyTableUrl"**) MultipartFile fre) **throws** IOException {  
 HttpSession session = request.getSession();  
 FlightControl flightControl = (FlightControl) session.getAttribute(**"flightControl"**);  
 **if** ((flightControl != **null**)) {  
 flightControl.setImageTransmission(imageTransmission);  
 session.setAttribute(**"flightControl"**, flightControl);  
 **return "itemInsert/FcIntergrat"**;  
 } **else** {  
 imageTransmission.setUseAlone(**true**);  
 session.setAttribute(**"imageTransmission"**, imageTransmission);  
 **return "itemInsert/ItIntergrat"**;  
 }  
 }  
  
 */\*\*  
 \* 响应图传路途请求  
 \* Insert it string.  
 \*  
 \** ***@param*** *session the HTTP servlet session  
 \** ***@return*** *the URI string  
 \*/* @Transactional(propagation = Propagation.REQUIRED)  
 @RequestMapping(value = **"InsertIt"**, method = RequestMethod.GET)  
 **public** String insertIt(HttpSession session) {  
 ImageTransmission imageTransmission = (ImageTransmission) session.getAttribute(**"imageTransmission"**);  
 imageTransmissionService.addImageTransmission(imageTransmission);  
 session.invalidate();  
 **return "itemInsert/InsertSuccessed"**;  
 }  
}