**仓储搬运设备远程故障诊断系统（BS服务器）详细设计说明书**

东南大学自动化学院

2016年11月22日

**版本历史**

|  |  |  |  |
| --- | --- | --- | --- |
| 版 本 号 | 更 改 描 述 | 更 改 日 期 | 修 订 人 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**目 录**

[1，引言 1](#_Toc468008175)

[1.1 编写目的 1](#_Toc468008176)

[1.2 项目背景 1](#_Toc468008177)

[1.3 术语定义及说明 2](#_Toc468008178)

[1.4参考资料 2](#_Toc468008179)

[2，设计概述 3](#_Toc468008180)

[2.1任务和目标 3](#_Toc468008181)

[2.2操作系统及软件配置 3](#_Toc468008182)

[2.3开发工具 3](#_Toc468008183)

[3，软件需求分析 4](#_Toc468008184)

[3.1软件功能需求分析 4](#_Toc468008185)

[3.2软件性能需求分析 4](#_Toc468008186)

[4，总体方案 5](#_Toc468008187)

[4.1框架设计 5](#_Toc468008188)

[4.2项目基本结构 5](#_Toc468008189)

[4.3应用程序包 6](#_Toc468008190)

[4.3.1 使用一个应用程序工厂 6](#_Toc468008191)

[4.3.2 在蓝图中实现应用程序的功能 7](#_Toc468008192)

[4.4启动脚本 9](#_Toc468008193)

[4.5需求文件 10](#_Toc468008194)

[4.6系统界面设计 11](#_Toc468008195)

[5，模块详细设计 16](#_Toc468008196)

[5.1用户认证模块 16](#_Toc468008197)

[5.1.1密码安全性 16](#_Toc468008198)

[5.1.2使用Flask-Login认证用户 16](#_Toc468008199)

[5.2数据库访问模块 17](#_Toc468008200)

[5.2.1定义模型 17](#_Toc468008201)

[5.2.2数据库操作 18](#_Toc468008202)

[5.3模板引擎 19](#_Toc468008203)

[5.4Bootstrap框架 19](#_Toc468008204)

[5.5highcharts图表库 20](#_Toc468008205)

[6，部署WEB服务器 21](#_Toc468008206)

[6.1安装Apache 21](#_Toc468008207)

[6.2安装Python 22](#_Toc468008208)

[6.3安装mod\_wsgi 22](#_Toc468008209)

# 1，引言

## 1.1 编写目的

本文档为 “基于物联网的仓储搬运设备远程故障诊断系统”中BS服务器软件的详细设计文档。经过认真的需求分析与系统论证之后确定了最终设计方案，包含网站实现的技术方案、项目的基本结构、用户认证、网站界面的设计、代理服务器的搭建等。

## 1.2 项目背景

随着工业技术的迅速发展，越来越多的自动化设备应用在了我们的日常生产生活当中，不断加快了工业化的进程。工业搬运车辆也被广泛应用在了港口、车站、机场、工厂车间、仓库、流通中心和配送中心等多个场所，叉车作为物料搬运设备中的主力军，在企业的物流系统中扮演着非常重要的角色。叉车主要发展于第二次世界大战期间，并于上世纪50年代逐渐开始在中国生产制造，而随着中国经济的快速发展，大部分企业的物料搬运工作已经脱离了原始的人工搬运模式，取而代之的是以叉车为主的机械化搬运。

仓储叉车是一种用于在仓库内搬运货物的叉车，其应用不仅能够节省人力物力，减少搬运的成本，还能进一步提高操作的安全性，避免事故的发生。大部分的仓储叉车都是以电动机驱动的，因其车体紧凑、移动灵活、自重轻和环保性能好等显著特点在仓储业中得到了广泛应用。

叉车等仓储搬运设备在使用过程中，不可避免的会出现各种类型的故障。现阶段生产过程中叉车出现故障后，一般还是采用传统的维修方法，即故障较为简单易诊断时，由用户自行在厂家指导下进行一些基础的维修工作；故障较为复杂难以诊断时，由厂家派出专业的维修人员前往现场对设备进行故障排除检查以及维修工作。厂家现场维护不仅会造成人力和物力上的浪费，增加维护成本，还会因为维修上的及时性影响用户的正常作业，导致生产无法正常运行。另一方面，当用户自行维护时，由于其对设备本身不够熟悉，也可能出现维护不当等问题，造成搬运设备的寿命下降。

物联网、云计算等技术的快速发展，为仓储搬运设备智能维护提供了新的解决方案。通过智能终端采集搬运设备操作和运行状态信息，并汇集到企业数据中心，通过故障分析与决策，可以实现搬运设备的远程监控，指导用户进行设备维护，从而可以提高搬运设备维护的智能化程度。因此，仓储搬运设备的远程监控的实现具有十分重要的意义。

## 1.3 术语定义及说明

文档中采用的专门术语的定义及缩略词简要如下：

**Python**：是一种面向对象的解释型计算机程序设计语言。

**Flask：**一个使用 Python 编写的轻量级 Web 应用框架。

**WSGI：**Web Server Gateway Interface，Web服务器网关接口。

**Werkzeug：**Flask的路由、调试和WSGI子系统。

**Jinja2：**Flask的模板系统。

**SQLAlchemy：**为高效和高性能的数据库访问设计，实现了完整的企业级持久模型。

**Apache：**最流行的Web服务器端软件。

**mod\_wsgi：**搭配Flask与Apache的Web服务器网关接口。

## 1.4参考资料

[1] 黄迪. 物联网的应用和发展研究[D].北京邮电大学,2011.

[2] 刘若冰. 物联网的研究进展与未来展望[J]. 物联网技术, 2011, 05:58-62.

[3] 周璞, 胡杰强, 龚军. 基于GPRS的电动汽车远程监控系统的开发设计: 第九届河南省汽车工程技术研讨会, 中国河南许昌, 2012[C].

[4] 朱炫鹏. 基于GPRS的远程监控系统的应用研究[D]. 东南大学,2005.

[5] 刘佳. 基于B/S结构的远程故障诊断系统研究与实现[D]. 南华大学, 2012.

[6] 尹洪珠. 基于Internet的机械设备的远程故障诊断中若干问题的研究[D]. 大连理工大学, 2002.

[7] 朱顺华. 基于GPRS的汽车远程故障诊断系统中车载终端的研究设计[D]. 国防科学技术大学, 2006.

# 2，设计概述

## 2.1任务和目标

BS服务器的目标是搭建稳定可靠的叉车监控网站，用户通过电脑浏览器即可进行访问，快捷方便。网站的内容包括：设备实时状态监控、设备故障实时报警及分析、历史故障信息统计与分析、历史状态信息及故障信息查询等。

BS服务器通过部署编写的web应用程序到代理服务器完成网站搭建，其中网站展示的数据来源为数据库中已有数据。

## 2.2操作系统及软件配置

服务器的操作系统为Windows Server 2008 R2 SP1（64位），Apache的版本为2.4，使用的WSGI版本为X64VC9(与Python及Apache版本匹配)。

## 2.3开发工具

综合考虑软件的开发效率及稳定性要求，拟采用Python下的Flask框架进行开发，数据库采用MySQL进行数据存储与查询。

# 

# 3，软件需求分析

## 3.1软件功能需求分析

用户通过访问BS服务器软件进行叉车监控，获取叉车信息。因此BS服务器软件需要提供全面充分的服务，功能需求主要包括以下几方面：

* 用户登录权限认证
* 在线叉车状态显示
* 故障报警显示
* 历史故障信息显示
* 历史状态信息查询
* 维修保养记录及查询
* 故障统计

## 3.2软件性能需求分析

BS服务器软件需要提供稳定的web服务，满足多个用户同时访问操作的需要，同时网站访问稳定可靠。经分析，服务器性能需求包括以下几方面：

* 能够长时间稳定运行，不能因为异常问题出现无法访问的问题。
* 访问网站速度快，不出现响应慢的问题。
* 满足多用户同时访问的问题，不互相影响。

# 4，总体方案

## 4.1框架设计

BS服务器软件的开发采用Python下的Flask框架，该框架属于小型框架，具有一个包含基本服务的强健核心，扩展性好，可以自己挑选所需的扩展包，组成一个没有附加功能的精益组合，从而完全精确满足自身要求。

Flask有两个主要依赖：路由、调试和Web服务器网关接口（WSGI）子系统由Werkzeug提供；模板系统由Jinja2提供。

Flask的数据库访问以扩展的形式实现，本项目中采用SQLAlchemy作为数据库框架，易用性强，与Flask集成度高，可以简化配置和操作。

## 4.2项目基本结构

Flask对大型项目没有特定的组织方式；应用程序的结构完全交给开发人员自己决定。虽然小型web应用程序用单个脚本可以很方便，但这种方法却不能很好地扩展。随着应用变得复杂，在单个大的源文件中处理会变得问题重重。因此一般采用以下方式来组织管理一个大型应用程序的包和模块。

|-flasky

|-app/

|-templates/

|-static/

|-main/

|-\_\_init\_\_.py

|-errors.py

|-forms.py

|-views.py

|-\_\_init\_\_.py

|-email.py

|-models.py

|-migrations/

|-tests/

|-\_\_init\_\_.py

|-test\*.py

|-requirements.txt

|-config.py

|-manage.py

这个结构有四个顶层目录：Flask应用一般放置在名为app的目录下；migrations目录包含数据库迁移脚本；单元测试放置在test目录下；还有一些新的文件：requirements.txt列出一些依赖包，这样就可以很容易的在不同的计算机上部署；config.py存储了一些配置设置；manage.py用于启动应用程序和其他应用程序任务。

## 4.3应用程序包

应用程序包放置了所有应用程序代码、模板和静态文件。它被简单的称为app，也可以给定一个特定于应用的名称(如果需要的话)。templates和static目录是应用的一部分，因此这两个目录应该放置在app中。数据库模型功能也要置入到这个包中，每个都以app/models.py形式存入自己的模块当中。

### 4.3.1 使用一个应用程序工厂

在单个文件中创建应用程序的方式非常方便，但是它有一个大缺点。因为应用程序创建在全局范围，没有办法动态的适应应用配置的更改:脚本运行时，应用程序实例已经创建，所以它已经来不及更改配置。对于单元测试这是特别重要的，因为有时需要在不同的配置下运行应用程序来获得更好的测试覆盖率。

解决这一问题的方法就是将应用程序放入一个**工厂函数**中来延迟创建，这样就可以从脚本中显式的调用。

这不仅给脚本充足的时间来设置配置，也能用于创建多个应用程序实例——一些在测试过程中非常有用的东西。这个构造函数导入大部分当前需要使用的扩展，但因为没有应用程序实例初始化它们，它可以被创建但不初始化通过不传递参数给它们的构造函数。create\_app()即应用程序工厂函数，需要传入用于应用程序的配置名。配置中的设置被保存在config.py中的一个类中，可以使用Flask的app.config配置对象的from\_object()方法来直接导入。配置对象可以通过对象名从config字典中选出。一旦应用程序被创建且配置好，扩展就可以被初始化。调用扩展里的init\_app()之前先创建并完成初始化工作。

app/ \_init\_\_.py：应用程序包构造函数如图1所示：



图1 ：应用程序包构造函数

### 4.3.2 在蓝图中实现应用程序的功能

应用程序工厂的转化工作引出了路由的复杂化。在单脚本应用中，应用程序实例是全局的，所以可以很容易地使用app.route装饰器定义路由。但是现在应用程序在运行时创建，app.route装饰器只有在create\_app()调用后才开始存在，这就太迟了。就像路由那样，这些通过app.errorhandler装饰器定义的自定义错误页面处理程序也存在同样的问题。

幸运的是Flask使用**蓝图**来提供一个更好的解决方案。一个蓝图就类似于一个可以定义路由的应用程序。不同的是，和路由相关联的蓝图都在休眠状态，只有当蓝图在应用中被注册后，此时的路由才会成为它的一部分。使用定义在全局作用域下的蓝图，定义应用程序的路由就几乎可以和单脚本应用程序一样简单了。

和应用程序一样，蓝图可以定义在一个文件或一个包中与多个模块一起创建更结构化的方式。为了追求最大的灵活性，可以在应用程序包中创建子包来持有蓝图。

app/main/ \_init\_\_.py：创建蓝图如图2所示。

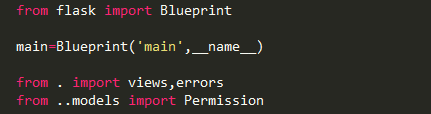


图2 ：创建蓝图

蓝图是通过实例化Blueprint类对象来创建的。这个类的构造函数接收两个参数：蓝图名和蓝图所在的模块或包的位置。与应用程序一样，在大多数情况下，对于第二个参数值使用Python的\_\_name\_\_变量是正确的。

应用程序的路由都保存在app/main/views.py模块内部，而错误处理程序则保存在app/main/errors.py中。导入这些模块可以使路由、错误处理与蓝图相关联。重要的是要注意，在app/init.py脚本的底部导入模块要避免循环依赖，因为view.py和errors.py都需要导入main蓝图。蓝图和应用程序一样注册在create\_app()工厂函数中。

访问网站放生403、404等错误时，根据错误类型调用相关函数进行处理返回。

 app/main/errors.py：蓝图的错误处理如图3所示。

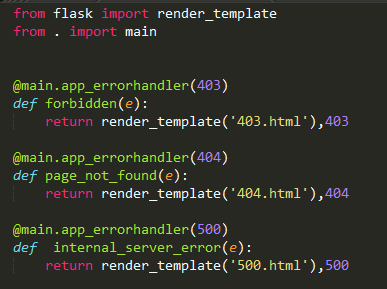


图3 ：蓝图的错误处理

正常访问网站时，根据路由不同，会调用相关函数进行处理返回。

app/main/views.py：带有蓝图的应用程序路由如图4所示。

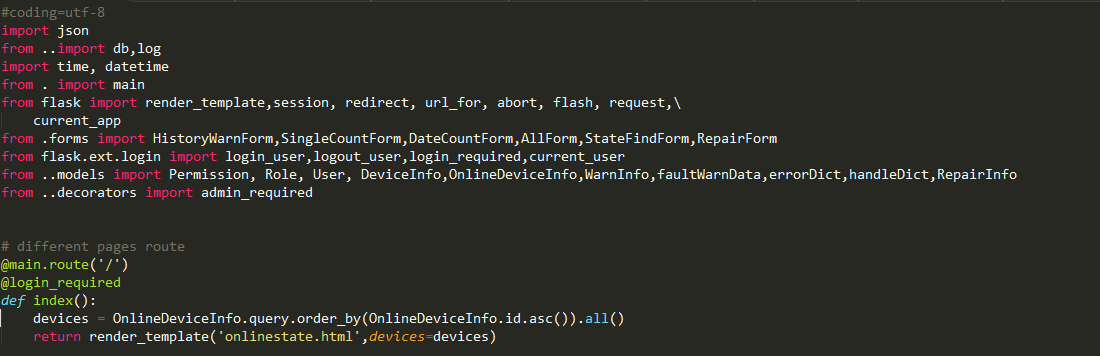


图4 ：带有蓝图的应用程序路由

访问根节点时（例如127.0.0.1）[,会触发图4中的main.route(‘/’)](mailto:会采用图4中的@main.route('/'))路由，调用相关函数，经过渲染后返回网页。

## 4.4启动脚本

顶层目录中的manage.py文件用于启动应用。这个脚本会在图5中展示。



图5 ：启动脚本

这个脚本开始于创建应用程序。使用环境变量FLASK\_CONFIG，若它已经定义了则从中获取配置；如果没有，则是用默认配置。然后用于Python shell的Flask-Script、Flask-Migrate以及自定义上下文会被初始化。

## 4.5****需求文件****

应用程序必须包含requirements.txt文件来记录所有依赖包，包括精确的版本号。这很重要，因为可以在不同的机器上重新生成虚拟环境，例如在生产环境的机器上部署应用程序。这个文件可以通过下面的pip命令自动生成：

pip freeze >requirements.txt

当安装或更新一个包之后最好再更新一下这个文件。以下展示了顶层目录中的requirements.txt，如图6所示。

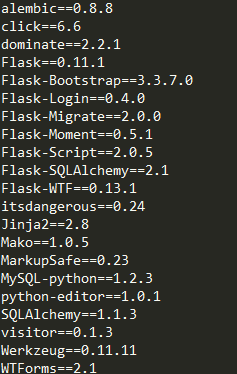


图6 ：requirements.txt需求文件内容

当你需要完美复制一个环境的时候，你可以运行以下命令创建一个新的环境：

pip install -r requirements.txt

## 4.6系统界面设计

根据模块功能对系统界面进行划分，系统分为以下几个界面。

* 实时在线叉车运行状态显示界面

该界面中显示在线叉车的状态显示，具体内容包括：

* 特定叉车查询区域（可以根据编号对叉车进行查询,显示满足特定条件的叉车）。
* 在线设备列表（显示满足i或ii条件的叉车列表，显示叉车基本信息，点击后可以查看叉车具体信息）。

实时在线叉车运行状态显示界面的设计如图7所示。



图7 实时在线叉车运行状态显示界面设计

* 叉车具体信息显示（在线叉车运行状态子界面）

该界面包含以下内容：

* 叉车基本信息（编号、运行时长等）。
* 行走控制器信息（数据来源为控制器can数据，包含电压、电流、温度、转速等）。
* 提升控制器信息（数据来源为智能终端采集的关于提升控制器的信息）。
* 喇叭等信息（数据来源为智能终端采集的非提升控制器部分的信息）。

叉车具体信息显示设计如图8所示。



图8 叉车具体信息界面设计

* 当日故障报警、历史故障显示界面

当日故障报警与历史故障报警类似，均包含故障显示列表，当日故障报警显示当日的故障，历史故障报警通过编号进行故障查询，界面包含以下内容：

* 故障显示列表
* 历史故障查询区域（可以根据叉车编号对故障进行查询）。

当日故障报警、历史故障显示界面设计如图9所示。



图9 当日故障报警、历史故障显示界面设计

* 故障报警具体信息显示（故障报警、历史故障子界面）

该界面包含以下内容：

* 叉车基本信息（编号）。
* 故障代码、发生次数。
* 故障信息名称、故障处理方法。
* 故障报警曲线。

故障报警具体信息显示设计如图10所示。



图9 故障报警具体信息显示界面设计

* 历史状态查询界面

该界面中显示历史状态信息，具体内容包括：

* 查询选择区域（选取需要查询的编号、时间范围）。
* 模拟量显示（以图表曲线的形式显示模拟量的变化）。
* 数字量显示（显示一定时间范围内的数字量（开关等）变化次数）。

历史状态查询界面设计如图11所示。



图11 历史状态查询界面设计

* 维修信息的添加及查询界面

该界面中显示维修信息的添加及查询信息，具体内容包括：

* 维修信息添加及查询
* 历史维修信息列表。

维修信息的添加及查询界面设计如图12所示。



图12 维修信息的添加及查询界面设计

维修保养清单添加包含以下内容：维修保养的叉车编号、维修时间、对应故障、维修策略、备注、维修人员。

维修保养清单添加界面设计如图13所示。



图13 维修保养清单设计

* 故障统计界面

该界面中显示故障统计信息，具体内容包括：

* 单辆叉车故障分布信息统计。
* 一定生产日期内叉车故障分布信息统计。
* 所有叉车故障分布信息统计。

故障统计界面设计如图14所示。



图14 故障统计界面设计

# 5，模块详细设计

## 5.1用户认证模块

网站要进行用户跟踪，用户访问网站时会进行身份认证，通过这一过程，程序可以获知用户是谁，就能提供有针对性的体验。最常用的认证方法就是要求用户提供一个身份证明（用户名）和一个密码。

本项目中用户认证使用到以下扩展包：

Flask-Login：管理已登录用户的用户会话。

Werkzeug：计算密码散列值并进行核对。

itsdangerrous：生成并核对加密安全令牌。

Flask-bootstrap：HTML模板。

Flask-wtf：Web表单。

### 5.1.1密码安全性

为了保证数据库中用户密码的安全，关键在于不能存储密码本身，而要存储密码的散列值。计算密码散列值的函数接收密码作为输入，使用一种或多种加密算法转换密码，最终得到一个和原始密码没有关系的字符序列。核对密码时，密码散列值可以代替原始密码，因为计算散列值的函数时可复现的：只要输入一样，结果就一样。

### 5.1.2使用Flask-Login认证用户

用户登录程序后，他们的认证状态要被记录下来，这样浏览不同的页面时才能记住这个状态，Flask-Login提供了这个功能，专门用来管理用户认证系统中的认证状态，且不依赖特定的认证机制。

Flask-Login在app/\_\_init\_\_.py程序的工厂函数中初始化，如图15所示。

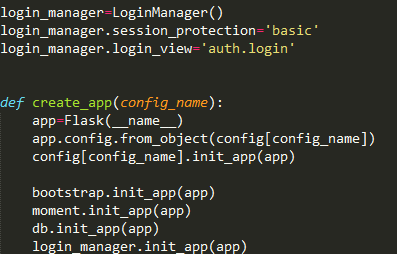


图15 Flask-Login初始化

为了保护路由只让认证用户访问，Flask-Login提供了一个login\_required修饰器，如果未认证的用户访问这个路由，Flask-Login会拦截请求，把用户发往登录页面。

本项目中提供用户登录、注册、修改密码、账户管理等功能。

## 5.2数据库访问模块

SQLAlchemy是一个很强大的关系型数据库框架，支持多种数据库后台（包含MySQL），并提供了高层ORM，易用性强。本项目使用Flask的扩展Flask-SQLAlchemy，该模块简化了在Flask中使用SQLAlchemy的操作。

## 5.2.1定义模型

模型这个术语表示程序使用的持久化实体。在ORM中，模型一般是一个Python类，类中的属性对应数据库表中的列。Flask-SQLAlchemy创建的数据库实例为模型提供了一个基类以及一系列辅助类和辅助函数，可用于定义模型的结构。

在app/model.py中定义数据库中访问的各种模型，以在线设备OnlineDeviceInfo为例，如图16所示。

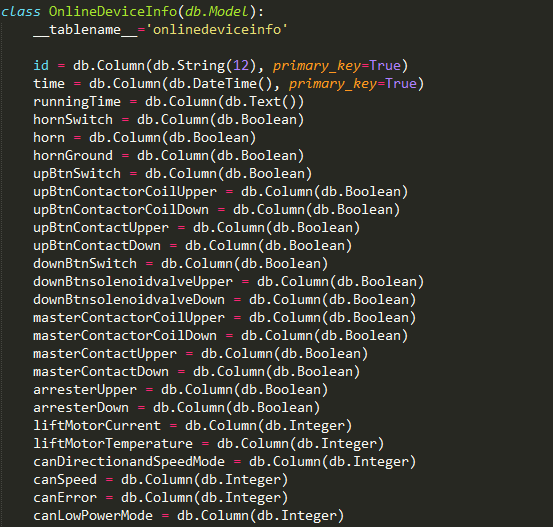


图16 OnlineDeviceInfo模型

## 5.2.2数据库操作

模型的构造函数接受的参数是作为关键字参数制定的模型属性初始值，通过数据库会话管理对数据库所做的改动，先要将其添加到会话中，为了将对象写入数据库，要调用commit()方法提交会话，示例如图17所示。



图17 数据库操作示例

Flask-SQLAlchemy为每个模型类都提供了query对象，可以配合过滤器进行更精确的数据库查询。

## 5.3模板引擎

模板是一个包含响应文本的文件，其中包含用占位变量表示的动态部分，其具体值只在请求的上下文才能知道。使用真实值替换变量，再返回最终得到的响应字符串，这一过程称为渲染。Flask使用了一个名为Jinja2的模板引擎。

Flask提供的render\_template函数把Jinja2模板引擎集成到了程序中，第一个参数是模板的名字，随后的参数是键值对，表示模板中变量对应的真实值，如图18所示。

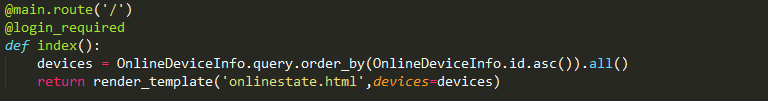


图18 后台渲染模板示例

模板文件中使用{{}}结构来表示变量，它是一种特殊的占位符，告诉模板引擎这个位置的值从渲染模板时使用的数据中获取，同时可以搭配使用循环、条件等多种控制语句，如图19所示。



图19 后台渲染模板示例

## 5.4Bootstrap框架

Bootstrap是Twitter开发的一个开源框架，它提供的用户界面组件可用于创建整洁且具有吸引力的网页，而且这些网页还能兼容所有现代Web浏览器。Bootstrap基于[HTML5](http://baike.baidu.com/view/951383.htm" \t "_blank)和[CSS3](http://baike.baidu.com/view/1713027.htm" \t "_blank)开发，在[jQuery](http://baike.baidu.com/view/1020297.htm)的基础上进行了更为个性化和人性化的完善，形成了一套自己独有的网站风格，同时兼容大部分jQuery插件。

通过引入 jquery.js、bootstrap.min.js 和 bootstrap.min.css 文件，可以让一个常规的 HTML 文件变为使用了 Bootstrap 的模板，如图20所示。



图20 引入Bootstrap相关文件

为标签的class属性添加Bootstrap中的类名，可以在模板中实例化Bootstrap组件，能够创建整洁且具有吸引力的网页，如图21所示。



图21Bootstrap使用示例

通过为class添加navbar等类名，可以实现导航栏的效果，如图22所示。



图22 Bootstrap实现效果

## 5.5highcharts图表库

Highcharts 是一个用纯[JavaScript](http://baike.baidu.com/view/16168.htm" \t "_blank)编写的一个图表库， 能够很简单便捷的在web网站或是web应用程序添加有交互性的图表，并且免费提供给个人学习、个人网站和非商业用途使用。HighCharts支持的图表类型有曲线图、区域图、柱状图、饼状图、散状点图和综合图表。

在模板引入 Highcharts相关文件可以在网页中实现有交互性的图表，引入export相关文件可以提供数据导出功能，如图23所示。



图23 引入Highcharts及export相关文件

HighCharts采用纯JavaScript编写，可以轻松实现具有交互性的图表，同时可以自定义图表属性，具体实现可以参照HighCharts官网教程，故不赘述。实现效果如图24所示。

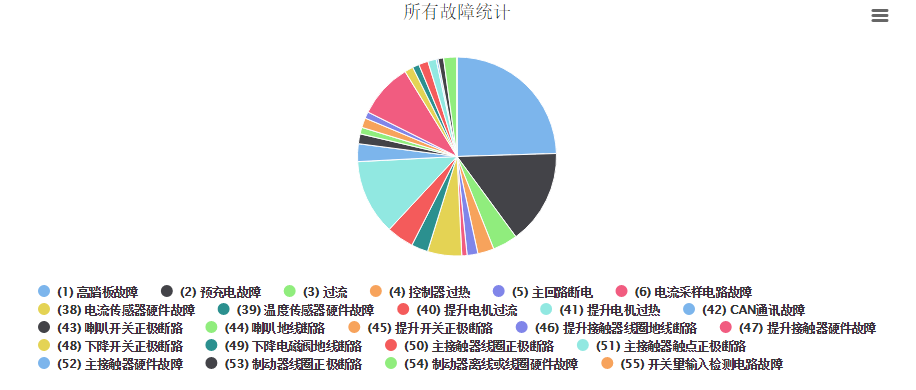


图24 HighCharts实现效果

# 6，部署WEB服务器

服务器的操作系统为Windows Server 2008 R2 SP1（64位），Apache的版本为2.4，Python版本为2.7.6，使用的mod\_wsgi版本为X64VC9(与Python及Apache版本匹配)。

## 6.1安装Apache

Apache是开源软件，针对windows环境，它不直接提供编译版本。可以在http://www.apachelounge.com/download/ 下载适合自己环境的版本。根据我自己当前的环境，我选择的是Win64 VC9编译出来的Apache2.4版本将压缩包下载到本机，然后将压缩包里面的Apache24文件夹拷贝到C:\。

接下来需要配置安装Apache的主服务，有了它，Apache才可启动：

打开CMD窗口，输入："C: \Apache24\bin\httpd.exe" -k install -n apache

服务安装完毕后可以打开浏览器，输入访问http://localhost 。若出现如图25所示界面，则Apache服务器的基本配置完毕。

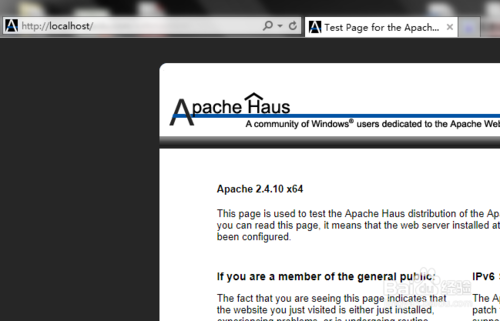


图25 Apache服务配置完成界面

同时，可以通过C:\Apache24\bin\ApacheMonitor.exe的文件管理Apache服务（提供start、stop、restart功能）。

## 6.2安装Python

通过安装包安装Python2.7.6后，需要安装Python相关模块，步骤如下：

1.下载https://bootstrap.pypa.io/ez\_setup.py

python ez\_setup.py #安装 setuptools

2.下载 https://raw.github.com/pypa/pip/master/contrib/get-pip.py

python get-pip.py install ##安装 pip

3、通过pip安装项目需求的模块

由于需求文件requirements.txt中包含所需模块，可以切换到该文件路径下，通过pip –install –r requirements.txt安装项目所需的模块。

## 6.3安装mod\_wsgi

下载的apache版本为2.4，python版本为2.7.6，下载对应于PYTHON2.7和Apache2.4的modwsgi，名称改为mod\_wsgi.so，添加到apache的moudle中，

然后对apache的conf下的httpd.conf进行配置，修改或添加内容如下：

<Directory />

    Options Indexes FollowSymLinks

    AllowOverride None

</Directory>

<VirtualHost \*:80>

        ServerName example.com

        WSGIScriptAlias / D:/ruyiweb/flasky/test.wsgi

        <Directory D:/ruyiweb/flasky>

Require all granted

        </Directory>

</VirtualHost>

其中test.wsgi 是自己写的wsgi脚本，里面内容如下：

import sys, os

sys.path.append(os.path.dirname("D:/ruyiweb/flasky/"))

from manage import app as application

到此为止，WEB服务器服务器部署完毕，通过浏览器即可访问项目网站，如图26所示。

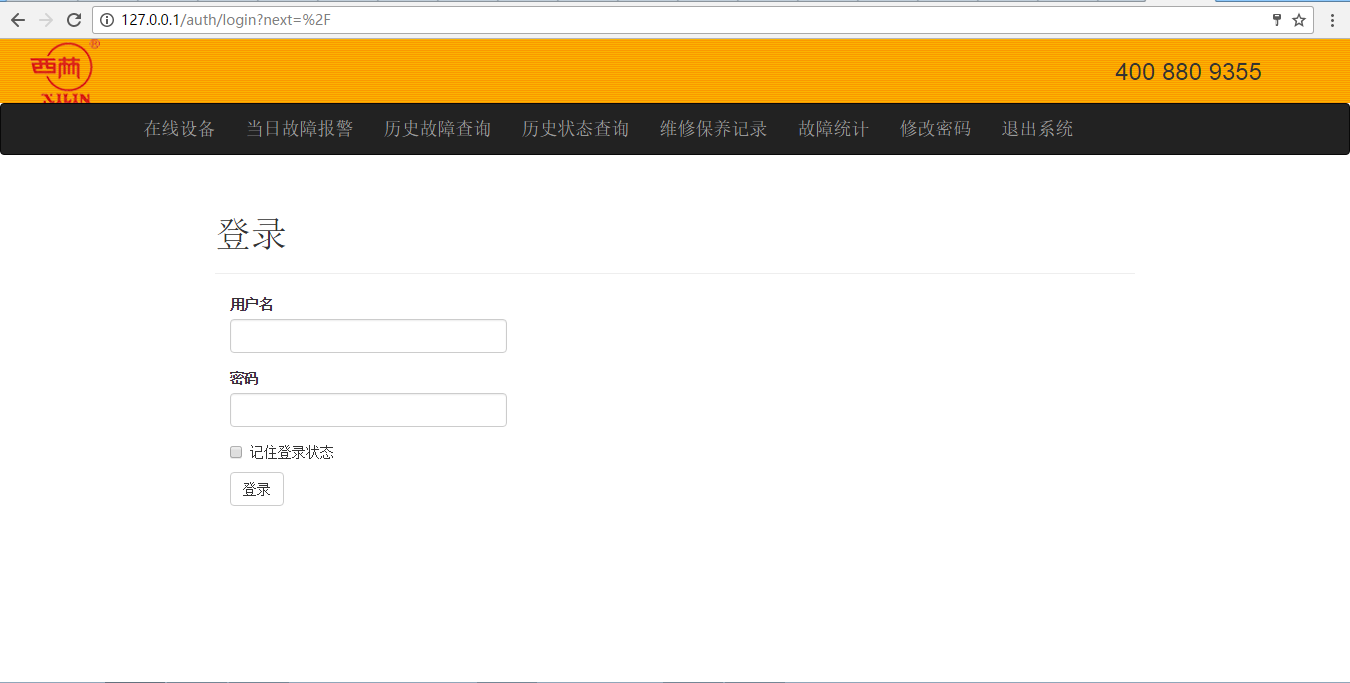


图26 部署完毕后访问网站