# **第三周：****数据库设计与实现**

### **注意：本节提供设计思路及范例代码引导你独立完成模块的实现。如果在代码实现中遇到问题，请在课程问答里提出。**

### **下载demo代码**

cd /home/shiyanlou/Code/

git clone http://git.shiyanlou.com/shiyanlou/cs354\_demo.git

## **一、实验简介**

本节实验为实验楼训练营课程《仿OpenStack开发云计算管理软件》第3节，希望与大家一起完成下面几项任务：

1. 数据库表格设计
2. SQLAlchemy对象实现
3. 数据库操作测试

本节内容相对简单，如果第二节实验还没有完成的同学可以在本周继续写虚拟化管理脚本。

## **二、Flask-SQLAlchemy**

我们使用的Flask是一个轻量级的Python Web框架，使用Flask构建一个大型的Web应用有很多细节，本课程主要参考文章中介绍的方法：[Flask: Large app how to](https://github.com/mitsuhiko/flask/wiki/Large-app-how-to)，推荐大家阅读。

在实验1中我们介绍过 Blueprint，每个 Blueprint 都是一个子模块，每个子模块包含下列的文件结构，我们以 user 模块举例：

loucloud/user/\_\_init\_\_.py

loucloud/user/views.py

loucloud/user/models.py

loucloud/user/constants.py

loucloud/user/forms.py

* \_\_init\_\_.py 文件将 user 模块可以公开的接口提供给其他组件。
* views.py 中提供 Flask 中的 URL 跳转以及返回数据。
* models.py 定义 user 数据库表。
* constants.py 含有 models.py 中所需的常量。
* forms.py 将基于 Flask-WTF 实现对象创建及修改的表单。

Flask-SQLAlchemy 组件是我们的应用与数据库连接的包，这个包中提供了一整套的类和函数方便我们进行数据库的创建以及读写操作。

在实验1中我们已经创建并初始化了 Flask-SQLAlchemy 对象，在 loucloud/extension.py文件中，本课示例代码cs354\_demo则放在dbs.py中，引用时需要注意：

from flask.ext.sqlalchemy import SQLAlchemy

db = SQLAlchemy()

后续数据库表的定义以及操作都将基于这个db对象。

## **三、数据库相关模块**

loucloud 中包含的核心数据表包括下面几个，我们分别命名对应的类：

* 用户：User
* 服务器：Host
* 镜像：Image
* 模板：Template
* 虚拟机：VirtualMachine

我们以服务器数据库定义举例，提供完整的服务器模块代码，大家可以参考以实现其他的组件。

在实现之前请到 /home/shiyanlou/Code/shiyanlou\_cs354 目录中初始化 virtualenv 的环境，同时使用 pip 安装依赖的软件包。初始化的方法已在实验1中已经学过。

### **3.1 服务器**

在实验2中我们介绍过服务器的操作包括添加和删除，用来为虚拟机提供物理资源。

我们先创建 host 模块目录及相关的文件列表：

cd /home/shiyanlou/Code/shiyanlou\_cs354

mkdir loucloud/host

touch loucloud/host/\_\_init\_\_.py

touch loucloud/host/constants.py

touch loucloud/host/forms.py

touch loucloud/host/models.py

touch loucloud/host/views.py

根据需求定义服务器表包含的数据项：

* 服务器IP地址或主机名：address
* 服务器SSH用户名：username
* 服务器SSH密码：password
* 服务器状态：status\_code (HOST\_OK，HOST\_ERROR)

根据上述需求，我们定义 host 模块数据库，并实现 loucloud/host/models.py 内容如下：

# -\*- coding: utf-8 -\*-

from sqlalchemy import Column

from ..dbs import db

from .constants import HOST\_OK

class Host(db.Model):

\_\_tablename\_\_ = 'hosts'

id = Column(db.Integer, primary\_key=True)

name = Column(db.String(128), nullable=False, unique=True)

username = Column(db.String(32), default="shiyanlou")

password = Column(db.String(32), default="shiyanlou")

status\_code = Column(db.SmallInteger, default=HOST\_OK)

注意我们使用到了 constants.py 文件中的常量HOST\_OK，因此在 constants.py 中我们需要定义如下内容：

# -\*- coding: utf-8 -\*-

# Host status

HOST\_OK = 0

HOST\_ERROR = 1

HOST\_STATUS = {

HOST\_OK: 'OK',

HOST\_ERROR: 'ERROR',

}

HOST\_STATUS 字典将用来在界面上根据服务器状态显示字符串。

我们如果要添加 host 模块相关的操作：查看，添加，删除，则需要在 views.py 定义操作的 url 及在 forms.py 中定义必要的表单，例如添加服务器所需要的表单。同时我们需要在 loucloud/templates/host/ 目录下增加必要的页面来展示。

首先 loucloud/host/views.py 中的内容：

#!/usr/bin/env python

# encoding: utf-8

from flask import Blueprint, render\_template, request, redirect, url\_for

from .forms import AddHostForm

from .models import Host

from ..dbs import db

host = Blueprint('host', \_\_name\_\_, url\_prefix='/hosts')

@host.route('/', methods=['GET', 'POST'])

def index():

if request.method == 'GET':

form = AddHostForm()

host\_list = Host.query.filter().all()

return render\_template("host/index.html", form=form, host\_list=host\_list)

else:

form = AddHostForm(request.form)

if form.validate\_on\_submit():

host\_instance = Host()

form.populate\_obj(host\_instance)

db.session.add(host\_instance)

db.session.commit()

return redirect(url\_for('host.index'))

@host.route('/<int:host\_id>/delete', methods=['GET'])

def delete\_host(host\_id):

host\_instance = Host.query.filter(Host.id==host\_id).first()

if host\_instance:

db.session.delete(host\_instance)

db.session.commit()

return redirect(url\_for('host.index'))

其次上面函数中用到的添加服务器的表单 `loucloud/host/forms.py:

#!/usr/bin/env python

# encoding: utf-8

from flask\_wtf import Form

from wtforms import (TextField, SubmitField, IntegerField)

class AddHostForm(Form):

name = TextField(u'Address')

username = TextField(u'Username')

password = TextField(u'Password')

status\_code = IntegerField(u'Status')

submit = SubmitField(u'Add')

再次我们的页面展示 loucloud/templates/host/index.html：

<!DOCTYPE html>

{% from "macros/\_form.html" import render\_form %}

<html lang="en">

<head>

<meta charset="UTF-8">

<title>host</title>

<script src="/static/js/jquery-1.11.3.min.js"></script>

<script src="/static/js/index.js"></script>

<script src="/static/bootstrap-3.3.5/js/bootstrap.min.js"></script>

<link rel="stylesheet" type="text/css" href="/static/bootstrap-3.3.5/css/bootstrap.min.css">

</head>

<body>

<nav class="navbar navbar-default">

</nav>

<div class="btn btn-success" data-toggle="modal" data-target="#myModal">+ Add Host</div>

<table class="table table-bordered table-striped table-hover">

<thead>

<tr>

<th>id</th>

<th>name</th>

<th>userame</th>

<th>password</th>

<th>status\_code</th>

<th>edit</th>

</tr>

</thead>

<tbody>

{% for item in host\_list%}

<tr>

<td>{{ item.id }}</td>

<td>{{ item.name }}</td>

<td>{{ item.username }}</td>

<td>{{ item.password }}</td>

<td>{{ item.status\_code }}</td>

<td>

<a href="{{ url\_for('host.delete\_host', host\_id=item.id )}}"><span class="btn btn-default">删除</span></a>

</td>

</tr>

{% endfor %}

</tbody>

</table>

<div class="modal fade" id="myModal">

<div class="modal-dialog">

<div class="modal-content">

<div class="modal-header">

<button type="button" class="close" data-dismiss="modal" aria-label="Close"><span aria-hidden="true">&times;</span></button>

<h4 class="modal-title">Add Host</h4>

</div>

<div class="modal-body">

{{render\_form(url\_for('host.index'), form)}}

</div>

</div>

</div>

</div>

</body>

</html>

上面的页面实现中我们使用了一个macros/\_form.html页面的render\_form函数，这个jinja函数实现的目的是将WTF的表单form展示成 HTML 页面表单。具体的内容很简单，完全用的是jinja的模板使用方法。

通过这几段示例代码我们已经完成了一个具备最基本功能的 host 模块，那只需要把 host 模块注册进 app 就可以进行测试了。host这个Blueprint如何注册是否还记得？如果忘记可以参考示例代码或者回顾第一节实验的代码。

启动测试：

由于实验环境中已经安装了MySQL，所以以下安装步骤可以略过。安装MySQL过程中，会设置root密码，请记住该密码。

sudo apt-get install mysql-server

实验楼环境中的MYSQL用户root，密码为空，这个用户我们后续会用到。

下面安装python-mysql

sudo apt-get build-dep python-mysqldb

下面为进入mysql控制台的命令，实验楼默认安装的mysql账号root的密码为空

sudo service mysql start

sudo mysql -uroot

进入了mysql控制台，然后执行

create user 'lc'@'localhost' IDENTIFIED BY '1qaz';

create database lc character set = utf8;

grant all on lc.\* to 'lc'@'localhost';

上述代码创建了用户1c，数据库lc，并将lc上的所有操作权限授予1c。注意上述步骤是配置loucloud使用的数据库用户信息，需要被写入Flask的config.py中SQLALCHEMY\_DATABASE\_URI，可参考示例代码中的配置。

后续步骤都在cs354\_demo下进行的示例操作，请注意把你自己的代码放在/home/shiyanlou/Code/shiyanlou\_cs354目录。

安装 python 依赖：

cd /home/shiyanlou/Code/cs354\_demo

virtualenv venv

source venv/bin/activate

pip install -r pip\_req/requirements.txt

初始化数据库，这里使用到的是flask.ext.migrate来对数据库进行版本控制，使用方法很简单，见manage.py中的示例代码，**每次当有数据库表项改动时都需要执行一次migrate和upgrade命令**。而init则是在数据库初始化时执行。

python manage.py db init

python manage.py db migrate

python manage.py db upgrade

运行程序

python run.py

打开环境中的浏览器，输入locahost:5000/hosts/，点击Add按钮，输入必要的信息添加服务器，添加后可以点击表单后的Delete按钮进行删除。

### **3.2 用户**

参考host模块实现 user模块的models.py文件。

用户数据库包含的项目：

* 用户名
* 用户密码（可以思考下如何能够避免保存用户密码）
* 用户邮箱
* 你可以想到的其他信息

### **3.3 镜像**

参考host模块实现 image 模块的models.py文件。

镜像数据库包含的项目：

* 镜像名
* 镜像文件路径

### **3.4 模板**

参考host模块实现 template模块的models.py文件。

模板数据库包含的项目：

* 模板名
* 关联的镜像ID（注意使用外键，可见Flask-SQLAlchemy文档数据表之间的关联方式）
* CPU数目
* 内存数目

### **3.5 虚拟机**

参考host模块实现 vm模块的models.py文件。

虚拟机数据库包含的项目：

* 虚拟机名
* 使用的模板ID（注意使用外键，可见Flask-SQLAlchemy文档数据表之间的关联方式）
* 虚拟机的所有者（关联到用户表的用户ID）
* 虚拟机的创建时间（使用SQLAlchemy的时间类型）

## **六、本节总结及下节预告**

本节实验学习了loucloud环境中所有数据库表的创建，示例程序中的代码仅供参考，可以多使用几种SQLAlchemy的数据类型。下一节基于我们这节实现的数据库封装成具体的操作逻辑，类似本节实现的服务器添加删除的功能。