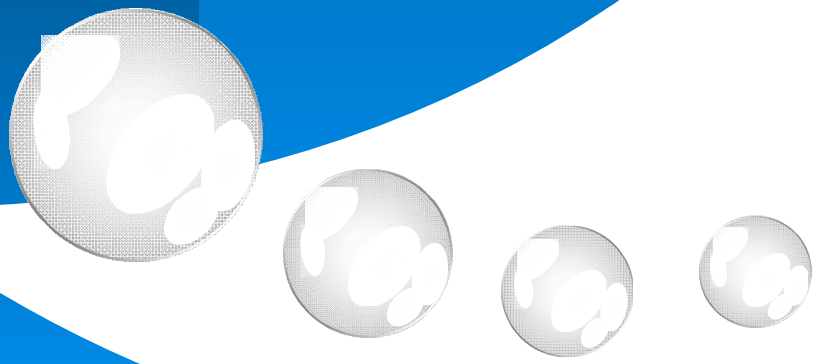
Web 框架开发思考与实

Uliweb



limodou@gmail.com

2011-12-4

个人介绍

- 2000 年开始学 Python，从 Zope 入手
- 2001 年担任过 LinuxForum.net 论坛的 Python 版发起人及版主
- 2004 年参与 HD 创建啄木鸟社区创建，同年开始开发 Ulipad
- 2005 年参与 CPUG 成立，多次参与 BPUG 的会课
- 2005 年编写过《 Django step by step 》
- 2008 年开始开发 Uliweb
- 写过很多技术博客， python-cn 邮件列表的创建者，参与以及创建过一些开源项目

几个问题

- 什么是框架？
- 什么是 Web 框架？
- Web 框架和 Web 应用有什么不同？
- 如何理解 Web 框架？

什么是框架？

- Martin Fowler
- http://martinfowler.com/bliki/InversionOfControl.html
- Inversion of Control is a key part of what makes a framework different to a library. A library is essentially a set of functions that you can call, these days usually organized into classes. Each call does some work and returns control to the client.
- A framework embodies some abstract design, with more behavior built in. In order to use it you need to insert your behavior into various places in the framework either by subclassing or by plugging in your own classes. The framework's code then calls your code at these points.

什么是 Web 框架

- 专注于 web 领域的框架
- 常见的 python web 框架：
  - 轻量级： web.py, bottle, flask
  - 中量级： django, web2py, uliweb, pyramid
  - 重量级： Zope?

Web 框架是一种实践

- Web 框架产生的原因就是把在 Web 开发中可以通用化，可以复用的内容，通过框架的形式进行固化，以便可以应用在其它的项目中。因此整个框架的设计就带有作者对 web 开发的理解烙印，它是思想的具体化。
- 因此也产生了不同的框架，不同的设计理念，如：wsgi 化， DRY 原则， URL 如何设计，模板设计等
- 一方面解决通用性的问题，另一方面又象大杂烩，只要能放到 web 开发中的，就有可能成为框架的一部分。

为什么会有全功能框架

- 因为问题域本身就可能是复杂和多变的
- 选择框架首先要看能不能解决问题域的大部分需求
- 随着发展，框架也在不断变得功能完整
  - 如： Web.py 原来也是单个文件，现在也由很多文件组成
  - 微框架解决了基本架构和功能，但是仍然有许多的扩展来实现其它的功能
- 复杂与全功能的区别（不讲复杂讲功能）
  - 复杂可能给我们感觉是做起来很麻烦或理解起来很困难
  - 功能多并不一定就是复杂的，这是两个维度的概念
  - 复杂产生的原因：问题域复杂，灵活性要求
- 全功能是一种缺省的实现，在简单的情况下，以最快的方式来实现所需要的功能。轻量框架在业务越来越复杂的情况下也在向全功能发展，但是有可能它是在应用层面的演变。

Web 框架和 Web 应用是不同的

- 目标不同：
  - Web 框架主要是为了解决 web 开发的效率和复用性等通用问题，它提供通用解决方案
  - Web 应用是为了解决个性化业务问题
- 应用范围不同：
  - Web 应用一般 >web 框架
  - Web 应用往往是多技术，软硬件结合的产物，如：异步、集群、分布式、并行计算
  - Web 框架的重点在于开发层面， Web 应用除了开发还涉及部署，运行管理等
  - 当一个功能在 web 框架中不提供时，两种扩展思路：
  - 应用层扩展（不具备很强的复用性）
  - 框架层扩展（考虑通用性和复用性）

Web 框架的基本结构

展示层

页面 | 报表展示 | 邮件 | 消息 | XMLRPC | Web Service | API

处理层

URL 处理 | View | Middleware | Signal | I18n | 日志管理

项目组织 | 配置管理 | Session | Cache | 用户认证 | 权限管理

服务层

ORM/NoSQL | 文件上传下载处理 | 静态文件

辅助

命令行工具 | 管理工具

常见 Web 框架都提供了些什么（一）

- 核心功能
  - 项目组织
  - 启动处理
  - 配置
  - URL Dispatch
  - Request, Response
  - View
  - Wsgi 服务
  - 信号扩展
  - 日志

常见 Web 框架都提供了些什么（二）

- 重要功能
  - Template
  - Form
  - ORM
  - I18n
  - Session
  - Cache
  - 命令行工具
  - 管理工具
  - 用户认证，权限
  - 中间件（ Middleware ）
  - …

常见 Web 框架都提供了些什么（三）

- 其它
  - XMLRPC, Web Service, API
  - RSS Feed
  - 邮件
  - 地图
  - Ajax 支持
  - 业务组件，如：论坛、 Blog 等

如何学习一个框架

- 把握住核心
- 了解设计理念
- 根据已有的经验，把感兴趣的内容深入下去
- 分析相同点，比较不同点
- 不断实践，甚至自行创建框架

Uliweb 的实践

- 08 年开始开发
- 目前最新的是 0.0.1 版
- 开发人员 1 人
- 目标：全功能 web 框架
- 设计原则：使用简单，重点在复用及配置化，方便扩展和替换

cn.pycon.org

Uliweb 的部署

- 支持标准的 wsgi 部署方式
- 目前尝试过在 dotcloud, sae 上部署，针对不同的环境分别提供相关的支持

Uliweb 的简单介绍

- 把我对 web 开发的理解及积累的一些经验通过框架形式固化下来，就形成了 Uliweb 。
- MVT(Model, View, Template) 模式
- APP 方式的项目管理
- 提供常见的功能，如： URL Dispatch, ORM, Template, View(Function or Class based), I18N, Mail，静态文件支持 , Form，文件上传下载 , Session, Cache
- 提供命令行工具
- 使用外部的核心组件： werkzeug, sqlalchemy
- Plugs 项目

最简单的 Hello, world 程序

- easy_install uliweb==0.0.1a6
- uliweb makeproject uliwebproject
- cd uliwebproject
- uliweb makeapp Hello
- uliweb runserver
- http://localhost:8000

```
#coding=utf-8
from uliweb import expose

@expose('/')
def index():
    return '<h1>Hello, Uliweb</h1>'
```

APP 形式的组织结构

- 最小的开发单位是 app
  - 它可以是任意可复用的内容，如：静态文件，模板， Views ， Models, Forms 等
  - 每个 app 是功能相对独立，资源独立，如： /static, settings.ini
- 提供 app 的依赖定义，使得引用相对简单
- 优点
  - 复用比较简单，基本上是自包含的，并且 app 下的 settings.ini 提供缺省配置信息

典型的目录结构

```
project/
    apps/
        settings.ini
        app1/
            templates/
            static/
            views.py
            models.py
            settings.ini
        app2/
    wsgi_handler.py
```

Python 语法的 ini 配置文件

- 采用文本形式的配置文件
- 基本样式为：

```
[GLOBAL]

INSTALLED_APPS = [

  'uliweb.contrib.orm'

]
```

- 重名变量的处理（可变的自动合并，不可变的替换）
- <= 强制替换
- Include 外部配置文件
- Settings.ini 处理顺序：
  - Default_settings.ini, app/settings.ini, project/settings.ini, project/local_settings.ini

功能配置 - 可替换，减少 HardCoding

- URL

[EXPOSES]

login = '/login', 'uliweb.contrib.auth.views.login'

- Signal

[BINDS]

audit.post_save = ‘post_save’

- FUNCTIONS

[FUNCTIONS]

flash = 'uliweb.contrib.flashmessage.flash‘

from uliweb import functions

functions.flash(‘message’)

- DECORATORS

Class-Based View

```
@expose('/user')
class UserView(object):
    def __begin__(self):
        if not request.user:
            return redirect('/login?next=%s' % request.path)

    @expose('/login')
    def login(self):
        #login process
        #URL = /login

    def register(self, name):
        #register process
        #URL = /user/register/<name>

    @expose('')
    def list(self):
        #URL = /user here '' will just user UserView class URL prefix '/user'

    def _common(self):
        #this function will not be exposed
```

可配置化 ORM

- 提供在 settings.ini 中对 Model 进行配置

```
[MODELS]

user = 'uliweb.contrib.auth.models.User'
```

- 在 Model 定义中可以使用字符串方式来引用 Model
- 提供 get_model() 来获得 Model 类
- 在 settings 中后定义的可以替换前面定义的 Model
- 去除象 from xxx import yyy, import yyy 这种 hard coding
- 有它使用的限制

资源链接的后台处理

- 通过提供 uliweb.contrib.template 提供 use 和 link 标签
- Use 标签提供了资源定义的简便方法
- 可以在每个 app 下定义一个 template_plugins 目录，下面包含将在 use 中使用的模块，在模块中定义要引用的资源链接，如：

```
a.append('pnotify/jquery.pnotify.default.css')

a.append('pnotify/jquery.pnotify.min.js')

return {'toplinks':a}
```

- 在页面渲染之后，将把链接与页面中的链接进行合并，去掉重复，并且可以指定 toplinks, bottomlinks 的位置

plugs/ui/jquery/jqutils/template_plugins/jqutils.py

```
def call(app, var, env, ajaxForm=False, hoverIntent=False,
spin=False):
    a = []
    a.append('jqutils/jqutils.css')
    if spin:
        a.append('jqutils/spin.min.js')
    a.append('jqutils/jqrselect.js')
    a.append('jqutils/jqutils.js')
    if ajaxForm:
        a.append('jqutils/jquery.form.js')
    if hoverIntent:
        a.append('jqutils/jquery.hoverIntent.minified.js')
    return {'toplinks':a, 'depends':[('jquery', {'ui':True})]}
```

原始的 **index.html**

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf8">
</head>
<body>
{{use "jqutils"}}
</body>
</head>
```

生成后的结果

```
<!DOCTYPE html>
<html>
<head>
<script type="text/javascript" src="/static/jquery/jquery-1.7.1.js"></script>
<link rel="stylesheet" type="text/css"
href="/static/jquery/ui/css/redmond/jquery-ui-1.8.16.custom.css"/>
<script type="text/javascript" src="/static/jquery/ui/js/jquery-ui-
1.8.16.custom.min.js"></script>
<script type="text/javascript"
src="/static/jquery/ui/js/jquery.ui.datepicker.zh.js"></script>
<link rel="stylesheet" type="text/css" href="/static/jqutils/jqutils.css"/>
<script type="text/javascript" src="/static/jqutils/jqrselect.js"></script>
<script type="text/javascript" src="/static/jqutils/jqutils.js"></script>
<meta charset="utf8">
</head>
<body>
</body>
</head>
```

Uliweb 文档

介绍:

- Uliweb 简介
- 安装说明
- 体系结构和机制
- 全局环境

教程:

- Simple Todo (Uliweb 版本) 之 基础篇
- Simple Todo (Uliweb 版本) 之 高级篇
- Hello, Uliweb
- 迷你留言板
- Sina Application Engine部署及开发指南

技术文档:

- Settings说明
- I18N 使用说明
- Mail 处理
- 部署指南
- 命令行工具使用指南
- 数据库和ORM
- 模板(Template)
- URL映射
- 视图(View)
- Form使用
- Middleware 开发
- Generic 说明
- 如何测试
- 日志处理说明
- Session使用说明
- XMLRPC 使用说明

Uliweb内置APP文档:

- staticfiles
- soap
- auth(用户认证处理)
- rbac(权限控制)
- template
- upload(文件上传及显示处理)

开发 Uliweb 的一些体会

- 用到了一些技术，如 ORM 中 metaclass, descriptor
- 造了许多的轮子，真正了解了不少的实现细节
  - Template
  - Form(layout 支持)
  - Request, response 的 local 绑定，与代理类的处理
  - I18n 的处理以及单词复数
  - 为什么要从 webob 换成 werkzeug
  - Weto 的创建
  - Html 代码使用 with 来处理
  - timezone
- 框架不是一成不变的，要学会吸收
- 坚持！

Uliweb 的一些资源

- 项目： code.google.com, github
- Plugs: code.google.com, github
- Uliweb-doc: github, uliweb.rtfd.org
- Uliweb uliweb.group

Q&A
Thank You!