中山大學
SUN YAT-SEN UNIVERSITY

本 科 生 毕 业 论 文

题　　目：REST 框架的设计与实现

院　　系：软件学院

专　　业：软件工程

学生姓名：高浩

学　　号：08386145

指导教师：赵淦森　　（教授）

二〇 一二 年　四 月

摘　要

目前在三种主流的 Web 服务实现方案中，因为 REST 模式的 Web 服务与复杂的 SOAP 和 XML-RPC 对比来讲明显的更加简洁，越来越多的 web 服务开始采用 REST 风格设计和实现。

REST 定义了一组体系架构原则，开发人员可以根据这些原则设计以系统资源为中心的 Web 服务，包括使用不同语言编写的客户端如何通过 HTTP 处理和传输资源状态。 如果考虑使用它的 Web 服务的数量，REST 近年来已经成为最主要的 Web 服务设计模型。事实上，REST 对 Web 的影响非常大，由于其使用相当方便，已经普遍地取代了基于 SOAP 和 WSDL 的接口设计。

通过理解 REST 的概念和原则，设计和实现一个 REST 框架的目的是为了减轻从头设计一款 REST 风格应用所需要的时间，从而可以快速、高效的创建一款 REST 风格的应用。REST 框架是把的实现将有助于减轻软件开发时一些具有共通性组件的重复开发，减轻了软件开发过程中的工作负荷，同时也提升了程序代码的重用性。

本文从设计到实现，介绍了如何构建一个 REST 框架，并给出了一个参考实现 RestPy3。通过 RestPy3 框架可以快速、高效的创建一款 REST 风格的应用。RestPy3 框架方便易用，提供和 Jersey 类似的 API，使用过 Jersey 的用户可以很快的学会如何使用。目前支持 Python 3 的 REST 框架十分少，现有的大部分框架都基于 Python 2 编写，如 Django Piston、Django REST framework 等，很多 REST 框架由于代码量大、依赖的类库比较多，所以一时间很难快速去支持 Python 3。RestPy3 由于是完全基于 Python 3 编写，所以在选择 Python 3 作为开发语言的人群中，还是有一定优势的。

关键词： REST；框架；Python；Web

Abstract

Currently there are three main solutions to build Web Services. RESTful Web Service is simpler than SOPA and XML-RPC. Therefore, more and more developers choose REST as their principle to build Web Services.

REST defines a set of architectural principles by which you can design Web services that focus on a system's resources, including how resource states are addressed and transferred over HTTP by a wide range of clients written in different languages. If measured by the number of Web services that use it, REST has emerged in the last few years alone as a predominant Web service design model. In fact, REST has had such a large impact on the Web that it has mostly displaced SOAP- and WSDL-based interface design because it's a considerably simpler style to use.

The purpose of design and implement the REST framework is to reduce the heavy burden to design a RESTful application from beginning to end. The implementation of the REST framework will help to reuse the common component of in the development and liberate the developer from hard work.

This thesis introduces how to design and implement a REST framework and gives a reference implementation, RestPy3. By using RestPy3, people can build a web application quickly and efficiently. RestPy3 is easy to use and provide Jersey-like API. People who are familiar with Jersey will use RestPy3 smoothly. Currently, few Rest Frameworks support Python 3. Most of them are based on Python 2, such as Django Piston and Django REST framework. It will take a long time when most of the frameworks migrate to Python 3 due to the complex dependence. People will find it is great to use RestPy3 when they choose Python 3 because RestPy3 is totally based on Python 3.

Keywords: REST; Framework; Python; Web

目　录

第一章　前言

1.1　项目的背景和意义

最近几年，Web API 已经改变了 Web 的面貌。从 2005 年开始，网站 Programmable Web 开始统计基于 SOAP 和 REST 的公开的 Web API。如图 1-1 所示，在 2005 年，Programmable Web 共统计到了 105 个 Web API，大部分来自 Amazon、Google、SalesForce 和 eBay。截止到 2008 年，能统计到的 Web API 的数量增长了 6 倍，达到 601 个，大部分公司开始发现把自己的数据开放给第三方所能带来的好处。截止到 2010 年末，开发人员已经有超过 2500 个 Web API 可以使用。Tesco 允许人们通过它提供的 Web API 购买商品；新浪微博提供的微博 API 允许开发人员访问微博用户的数据；人人网提供的 Web API 同样允许开发人员访问它的数据。截止到 2011 年，可供使用的 Web API 已经超过了 5000 个，并且出现了很多使用这些 Web API 所构建出来的杀手级的应用。[1]

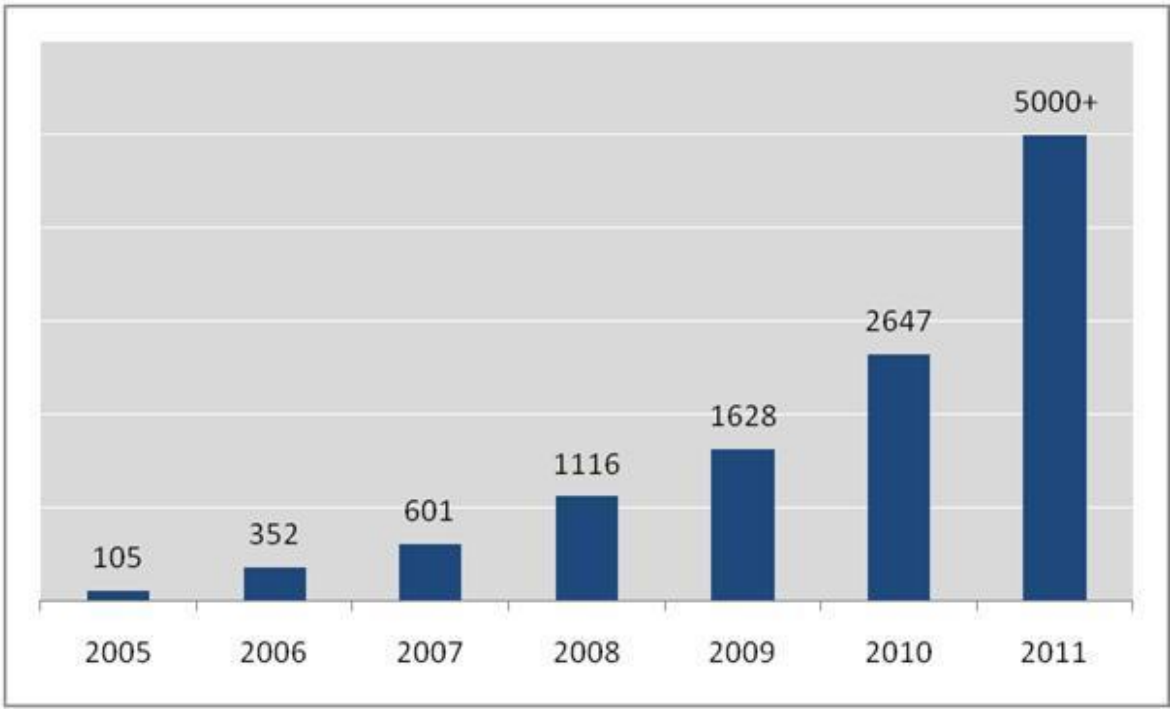

图 1-1：可用 Web API 逐年递增图[1]

[1] 图片引用自 www.**programmableweb**.com/

过去的十年中，遵从 SOA（面向服务的架构）思想的指导下的企业级应用各个模块之间实现了解耦，同时提供了规范的服务接口，从而使系统各个模块的重用成为可能。SOA 的种种好处使得 SOA 受到了大力追捧，但是很少真正有公司真正的实践了 SOA 的原则。与此同时，Web 却已经变成了一个主要的面向服务的平台，在 Web 上，大量的信息和丰富的服务可以通过 Web API 访问。很多 Web API 一开始是基于 SOAP 的，然而现在，趋势在改变，并且向着 REST 靠拢，最新的 Web API 大部分都是 REST 风格的。REST 风格的 Web API 正在迅速的增长，增势图，如图 1-2 和 1-3 所示。

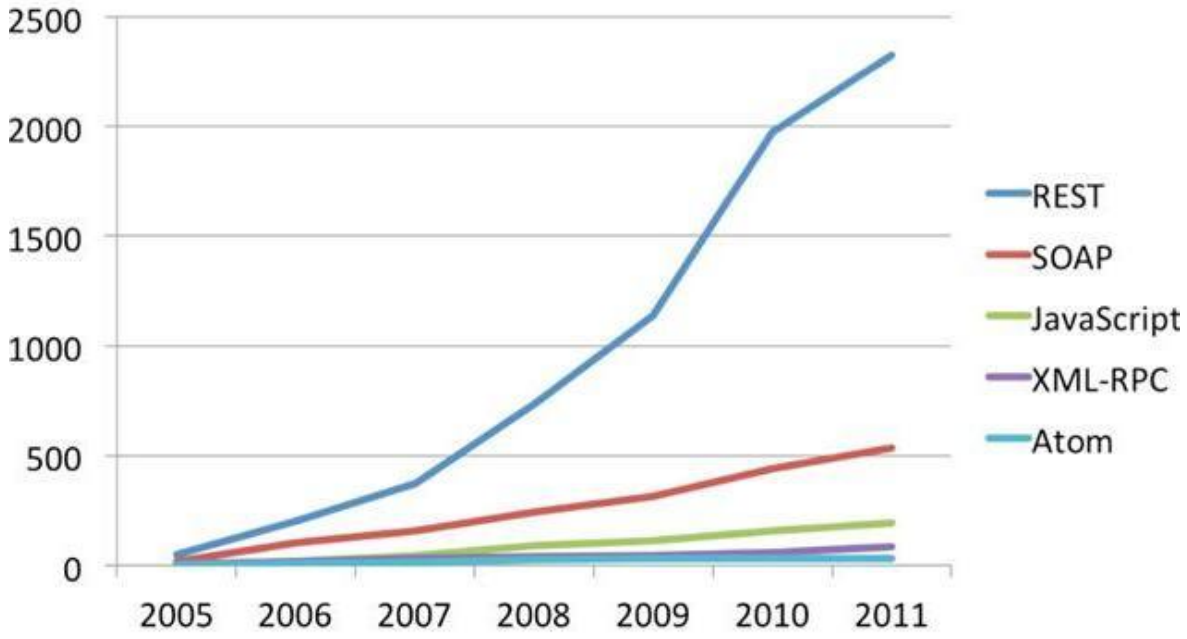

图 1-2：REST 风格 Web API 增势图[2]

[2] 图片引用自 www.**programmableweb**.com/

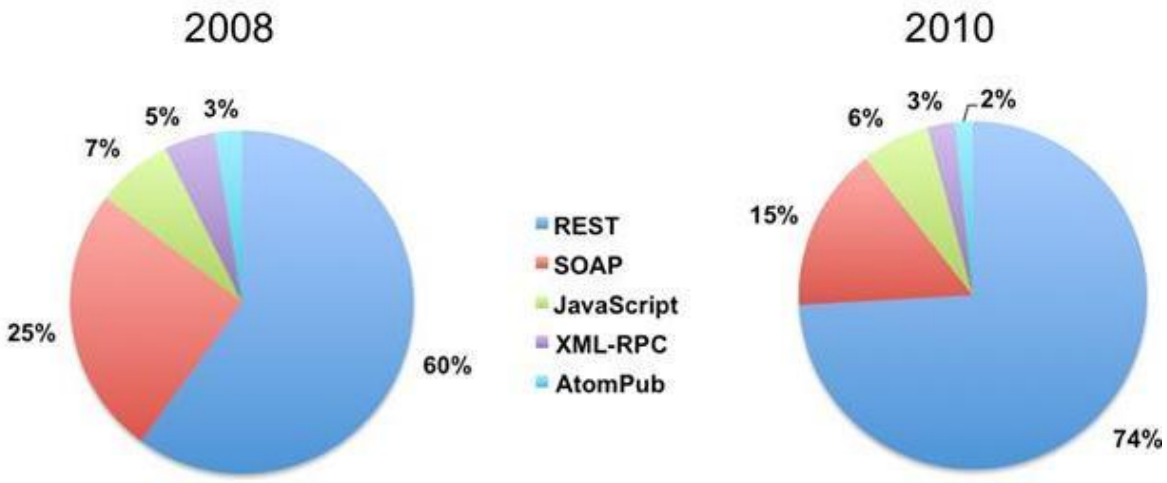

图 1-3：REST 风格 Web API 所占比重图[3]

1.2 研究开发现状分析

目前存在个各种各样的 REST 框架的实现。例如，REST 框架的 Java 语言实现有 Jersey、Restlet、Apache CXF、等。其中，Jersey 是 JSR311（JAX-RS）规范的参考实现。尽管 REST 在 2000 年已经被提出，且在 Java 社区中，早已有 Restlet 和 RestEasy 之类的架构，可以帮助开发人员方便的实现 RESTful 的 Web 服务，但是他们不够直观。直到 2008 年 JSR311 规范制定后，才将其标准化，并在之后又 Sun 公司公布了第一个参考实现。

REST 框架的 Python 语言实现有 Django Piston、Django REST framework 等。REST 框架的 Python 语言实现相对来说较少，加上 Python 3 和 Python 2 的不兼容，导致许多早期的 REST 框架的实现依然保留在 Python 2 的阶段，一段时间内也无法升级到 Python 3。例如，由于 Django 目前还没有推出支持 Python 3 的版本，所以像是 Django Piston 或是 Django REST framework 之类的 REST 框架目前也没有支持 Python 3 的意向。这导致一定程度上，基于 Python 3 的 REST 框架更是少之又少。

[3] 图片引用自 www.**programmableweb**.com/

1.3 项目的目标和范围

本项目旨在构建一个基于 Python 3 的轻量级的 REST 框架 RestPy3，使用户方便、快速、高效的创建 REST 风格的应用。

项目的目标和特性：

- 提供健壮的 RESTful 接口，提高开发速度，降低开发复杂度。
- 严格的遵循 WSGI 接口，Web 服务器可以选择任何遵循 Python 服务器网关接口开发的服务器。
- 提供类似于 JAX-RS 风格的 API，通过标注的方式来构建应用，方便使用。

1.4 论文结构简介

本文共分为 章，结构和各章的主要内容组织如下：

第一章介绍了 Web API 的现状和流行框架的使用比例和现状、着重介绍了 REST 框架的现状、本文的研究工作及组织结构。

第二章介绍了框架实现所涉及到的相关理论，通过对 REST 的概念、JSON、正则表达式、Python 服务器网关接口的介绍，为框架具体设计和实现，做了铺垫。

第三章介绍了需求分析，并进行了需求建模。通过用例图来明确需求。

第四章介绍了框架实现的所用到的架构的原理和方法。

第五章介绍了框架实现的各个模块和框架实现中比较重要的类，给出了 RestPy3 框架的具体实现。

第六章介绍了 RestPy3 框架的使用、运行的前提条件，并给出了示例，如何利用 RestPy3 框架快速建立一个 REST 风格的 Web 应用。

第七章是对本文内容总结与展望。

第二章　技术与原理

构建一个 REST 框架需要用到很多技术，本章介绍了实现 RestPy3 所使用的技术及其原理，为后面深入讲解 RestPy3 的实现提供了理论依据。

2.1　Representational State Transfer（代表性状态转移）

REST 这个概念于 2000 年由 Roy Fielding 在就读加州大学欧文分校期间在学术论文“Architectural Styles and the Design of Network-based Software Architectures”首次提出，他的论文中对使用 Web 服务作为分布式计算平台的一系列软件体系结构原则进行了分析，而其中提出的 REST 概念并没有获得现在这么多关注。 多年以后的今天，REST 的主要框架已经开始出现，但仍然在开发中，因为它已经被广泛接纳到各个平台中。[2] [3] [4]

2.1.1 REST 定义

REST 定义了一系列服务架构准则。通过这些准则，你可以设计以系统资源为中心的 Web 服务，包括使用不同语言编写的客户端如何通过 HTTP 处理和传输资源状态。

REST 在 Web 领域已经得到了广泛的接受，是基于 SOAP 和 Web 服务描述语言（Web Services Description Language，WSDL）的 Web 服务的更为简单的替代方法。 接口设计方面这一转变的关键证据是主流 Web 2.0 服务提供者（包括 Yahoo、Google 和 Facebook）对 REST 的采用，这些提供者弃用或放弃了基于 SOAP 和 WSDL 的接口，而采用了更易于使用、面向资源的模型来公开其服务

2.1.2 REST 风格的 Web 服务

RESTful Web 服务（也称为 RESTful Web API）是一个使用 HTTP 并遵循 REST 原则的 Web 服务。它从以下三个方面资源进行定义：[5][6][7]

1. URI，比如：http://example.com/resources/。
2. Web 服务接受与返回的互联网媒体类型，比如：JSON，XML ，YAML 等。
3. Web 服务在该资源上所支持的一系列请求方法（比如：POST，GET，PUT

或 DELETE）。

表 2-1 列出了在实现 RESTful Web 服务时 HTTP 请求方法的典型用途。

表 2-1： HTTP 请求方法及其用途表

资源	GET	PUT	POST	DELETE
一组资源的 URI，比如 http://example.com/resources/	列出 URI，以及该资源组中每个资源的详细信息（后者可选）。	使用给定的一组资源替换当前整组资源。	在本组资源中创建/追加一个新的资源。该操作往往返回新资源的 URL。	删除整组资源。
单个资源的 URI，比如 http://example.com/resources/142	获取指定的资源的详细信息，格式可以自选一个合适的网络媒体类型（比如：XML、JSON 等）	替换/创建指定的资源。并将其追加到相应的资源组中。	把指定的资源当作一个资源组，并在其下创建/追加一个新的元素，使其隶属于当前资源。	删除指定的元素。

对于 REST 风格的 Web 服务，其具体实现应该遵循四个基本设计原则：

1. 显式地使用 HTTP 方法。基于 REST 的 Web 服务的主要特征之一是以遵循 RFC 2616 定义的协议的方式显式使用 HTTP 方法。REST 要求开发人员显式地使用 HTTP 方法，并且使用方式与协议定义一致。这个基本 REST 设计原则建立了创建、读取、更新和删除（create, read, update, and delete，CRUD）操作与 HTTP 方法之间的一对一映射。
2. 无状态。REST Web 服务应用程序（或客户端）在 HTTP Header 和请求正文

中包括服务器端组件生成响应所需要的所有参数、上下文和数据。这种意义上的无状态可以改进 Web 服务性能，并简化服务器端组件的设计和实现，因为服务器上没有状态，从而消除了与外部应用程序同步会话数据的需要。

3. 公开目录结构式的 URI。从对资源寻址的客户端应用程序的角度看，URI 决定了 REST Web 服务将具有的直观程度，以及服务是否将以设计人员能够预测的方式被使用。REST Web 服务 URI 的直观性应该达到很容易猜测的程度。将 URI 看作是自身配备文档说明的接口，开发人员只需很少（如果有的话）的解释或参考资料即可了解它指向什么，并获得相关的资源。为此，URI 的结构应该简单、可预测且易于理解。
4. 传输 XML、JavaScript Object Notation (JSON)，或同时传输这两者。资源表示形式通常反映了在客户端应用程序请求资源时的资源当前状态及其属性。这种意义上的资源表示形式只是时间上的快照。 这可以像数据库中的记录表示形式一样简单，其中包括列名称与 XML 标记之间的映射，XML 中的元素值包含行值。 或者，如果系统具有数据模型，那么根据此定义，资源表示形式是系统的数据模型中的对象之一的属性快照。 这些对象就是您希望您的 REST Web 服务为客户端提供的资源。

2.1.3 REST 与 SOAP 对比

SOAP 是一个在去中心化、分布式环境下的轻量级的信息交换协议。它通过 XML 技术来定义一个可扩展的信息交换机制，提供构建信息的方式，从而使信息可以在各种协议下传输。这个架构被设计为不依赖于任何特殊的编程模型和具体语义的实现。**[8]**

典型的基于 SOAP 的 Web 服务以操作为中心，每个操作接受 XML 文档作为输入，提供 XML 文档作为输出。在本质上讲，它们是 RPC 风格的。而在遵循 REST 原则的 ROA 应用中，服务是以资源为中心的，对每个资源的操作都是标准化的 HTTP 方法。**[9][10]**

REST 的优点如下：

1. 可以利用缓存 Cache 来提高响应速度。
2. 通讯本身的无状态性可以让不同的服务器的处理一系列请求中的不同请求，

提高服务器的扩展性。

3. 浏览器即可作为客户端，简化软件需求。
4. 相对于其他叠加在 HTTP 协议之上的机制，REST 的软件依赖性更小。
5. 不需要额外的资源发现机制。
6. 在软件技术演进中的长期的兼容性更好。

相对于 SOAP，REST 还有以下优点：

1. 基于 REST 构建的系统其系统的扩展能力要强于 SOAP，这可以体现在它的统一接口抽象、代理服务器支持、缓存服务器支持等诸多方面。并且，伴随着 Web Site as Web Services 演进的趋势，基于 REST 设计和实现的简单性和强扩展性。
2. 不像基于 SOAP 的 Web 服务，RESTful Web 服务并没有的“正式”标准。[7] 这是因为 REST 是一种架构，而 SOAP 只是一个协议。虽然 REST 不是一个标准，但在实现 RESTful Web 服务时可以使用其他各种标准（比如 HTTP，URL，XML，PNG 等）。

2.2 JSON(JavaScript Object Notation)

JSON(JavaScript Object Notation) 是一种轻量级的数据交换格式。易于人阅读和编写。同时也易于机器解析和生成。它基于JavaScript Programming Language，Standard ECMA-262 3rd Edition - December 1999的一个子集。 JSON 采用完全独立于语言的文本格式，但是也使用了类似于 C 语言家族的习惯（包括 C, C++, C#, Java, JavaScript, Perl, Python 等）。 这些特性使 JSON 成为理想的数据交换语言。[11]

REST 之所以受到欢迎的一个原因是它有更好的客户端可达性。SOAP 是为企业级应用设计的，协议设计的时候充分考虑了平台不可知性。所以，SOAP 的 XML 是冗长的，而且通常使用起来很麻烦。JSON 则因为它精简的表达方式而更加易读，它还是 JavaScript 原生支持的格式。

图 2-1 可以反映出支持 JSON 的 Web API 的数量逐年递增，截止到 2010 年，已经有将近 45%的 Web API 支持 JSON 格式了，更有一些 Web API 则只支持 JSON 格式。

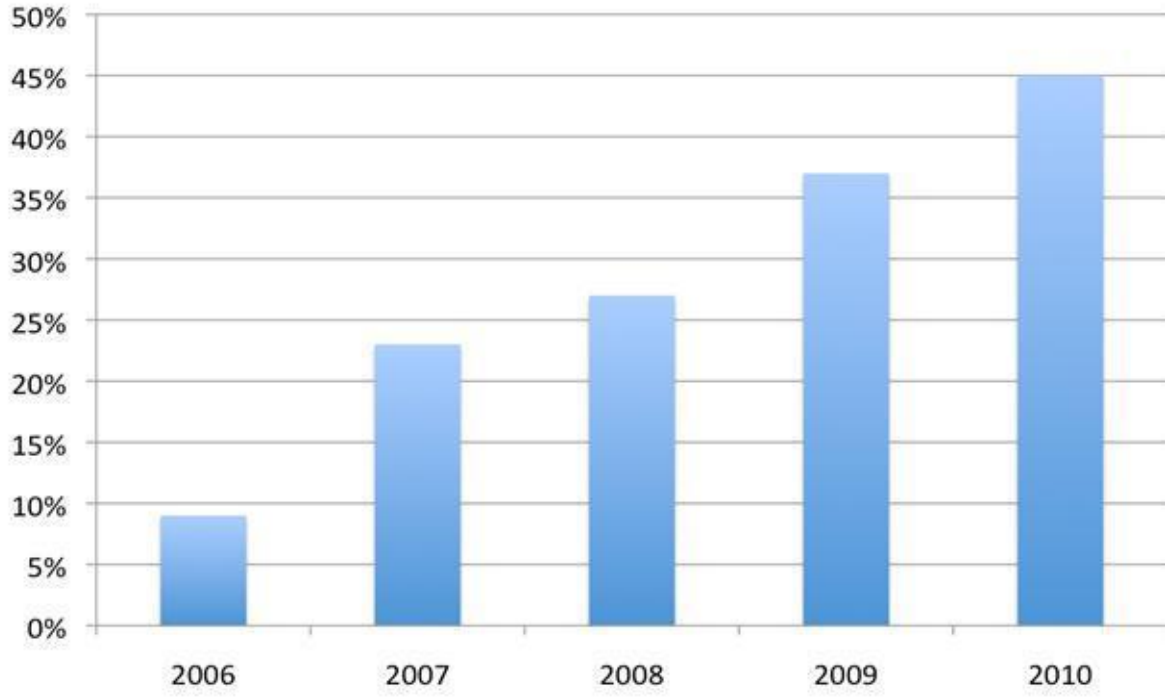

图 2-1：支持 JSON 格式的 Web API 逐年递增图[4]

2.2.1 JSON 的结构

JSON 建构于两种结构：

1. “名称/值”对的集合（A collection of name/value pairs）。不同的语言中，它被理解为对象（object），纪录（record），结构（struct），字典（dictionary），哈希表（hash table），有键列表（keyed list），或者关联数组 （associative array）。
2. 值的有序列表（An ordered list of values）。在大部分语言中，它被理解为数组（array）。

这些都是常见的数据结构。事实上大部分现代计算机语言都以某种形式支持它们。这使得一种数据格式在同样基于这些结构的编程语言之间交换成为可能。

2.2.2 JSON 和 XML 对比

JSON 的优点如下：

1. 基于纯文本，跨平台传递极其简单。
2. Javascript 原生支持，后台语言几乎全部支持

[4] 图片引用自 www.**programmableweb**.com/

3. 轻量级数据格式，占用字符数量极少，特别适合互联网传递；
4. 可读性较强，虽然比不上 XML 那么一目了然，但在合理的依次缩进之后还是很容易识别的；
5. JSON 在编码和解码上都比 XML 简单许多。XML 有丰富的编码工具，比如 Dom4j、JDom 等，JSON 也有 json.org 提供的工具，但是 JSON 的编码明显比 XML 容易许多，即使不借助工具也能写出 JSON 的代码，可是要写好 XML 就不太容易了

JSON 与 XML 最大的不同在于 XML 是一个完整的标记语言，而 JSON 不是。这使得 XML 在程式判读上需要比较多的功夫。主要的原因在于 XML 的设计理念与 JSON 不同。XML 利用标记语言的特性提供了绝佳的延展性（如 XPath），在数据存储,扩展及高级检索方面具备对 JSON 的优势，而 JSON 则由于比 XML 更加小巧,以及浏览器的内建快速解析支持,使得其更适用于网络数据传输领域。[12][13][14][15]

2.3 正则表达式和 URL 映射

正则表达式是一个描述特定的搜索模式的特殊的文本字符串。正则表达式通常被用来检索、替换那些符合特定模式的文本内容。[16]

一个 Web 应用框架需要有一个 URL 映射机制，而其中最重要的就是 URL 的匹配，即字符串的匹配。正则表达式提供了灵活、高效的字符串匹配方法。

Web 应用框架中的 URL 映射就是一种翻译 URL 的机制，目前常见的一共有三种 URL 映射方法。

1. 模式匹配（pattern matching），一些框架通过正则表达式，使用预先设定的模式来匹配提供的 URL，像是 Drupal 和 Django。
2. URL 重写（URL rewriting），而另一些则通过 URL 重写来把 URL 转换成潜在的搜索引擎可以识别的模式。
3. 图遍历（graph traversal），这种技术则是使用图的遍历的方法，通过把 URL 分解成遍历图的步骤。使用这种技术的有 Zope。

URL 映射系统通过使用模式匹配或是 URL 重写，可以使得 URL 更加易用，使网站的结构更加简洁，除此之外，还能使搜索引擎更加容易对网站进行索引。比如，要显示在计算机类别下面所有有关 Web 的图书，原来的 URL 是"/books?cat=computer&topic=web"，通过 URL 映射，可以被简单的替换成"/books/computer/web"。通过 URL 映射，使得 URL 更加易读，而且提供给搜索引擎更加结构化的站点信息。

图遍历的方法也可能会使得 URL 更加简单易读，比如，"/ books/web "很可能代表的是"/ books/computer/web"，因为"/ books/web "的图遍历结果可能就是"/books/computer/web"。

在Python中内建了一个功能强大的正则表达式引擎，通过 Python 提供的正则表达式库，可以方便高效的匹配字符串，从而实现 URL 映射。所以，RestPy3 使用模式匹配的方法，使用正则表达式，来实现 URL 映射系统。

2.4 Python Web Server Gateway Interface(服务器网关接口)

Python 目前拥有大量的 Web 编程框架，例如 Zope, Quixote, Webware, SkunkWeb,

PSO 和 Twisted Web。大量的选择对于新手来说是个问题，因为一般来说，Web 框架的选择会限制 Web 服务器的选择。

相比之下，虽然 Java 也拥有众多的 Web 框架，但是 java 的 servlet API 使得基于任何 Java Web 框架编写的应用程序可以在任何支持 servlet API 的 Web 服务器上运行。在 Python 中使用这种类似的 API（不管服务器是用 Python 写的，还是内嵌 Python，还是通过一种协议来启动 Python）的使用和普及，将分离人们对 Web 框架和对 Web 服务器的选择，用户可以自由选择适合他们的组合，而 Web 服务器和 Web 框架的开发者也能够把精力集中到他们各自擅长的领域中。[17]

PEP (Python Enhancement Proposals) 3000 中建议在 Web 服务器和 Web 应用或 Web 框架之间建立一种简单、通用的接口规范：Python Web Server Gateway Interface (WSGI)。如图 2-2 所示，任何实现了这个接口规范的 Web 应用或者 Web 架构都可以在实现了这个接口规范的 Web 服务器上运行。

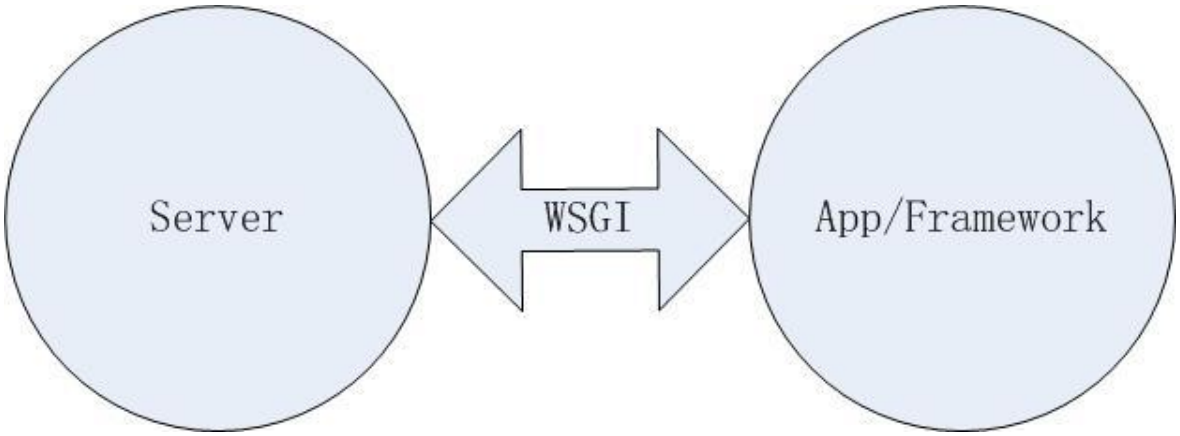

图 2-2：WSGI 接口示意图

所以，在框架这一端，我么需要实现这个接口，从而使得框架的应用范围更加广。

实现这个接口十分简单，我们只需要定义一个接受两个参数的应用程序对象，例如：

```
class WSGIApplication(object):

    def __call__(self, environ, start_response):
        response_body = 'hello world!'
```

```
        status = '200 OK'
        response_headers = [('Content-Type', 'text/plain'),
                            ('Content-Length', str(len(response_body)))]
        start_response(status, response_headers)
        return [response_body]

    def run(self, host, port, make_server=None):
        if not make_server:
            from wsgiref.simple_server import make_server
        server = make_server(host or '127.0.0.1', port, self)
        from contextlib import closing
        with closing(server.socket):
            server.serve_forever()
```

其中 WSGIApplicatio 实现了服务网关接口的应用程序对象。一个应用程序对象是一个简单的接受两个参数的可调用对象，这里的“对象”所指的并不是真的需要一个对象实例，一个函数、方法、类、或者带有 __call__ 方法的对象实例都可以用来做应用程序对象。应用程序对象必须可以多次被请求，事实上，服务器/网关(而非 CGI)确实会产生这样的重复请求。

虽然我们把上面用到“应用程序”对象这个术语，但并不是指我们要把 WSGI 当作 API 来调用，我们假定应用程序开发者仍然使用更高层面上的框架服务来开发应用程序，WSGI 是提供给框架和服务器开发者使用的工具，并不打算直接对应用程序开发者提供支持

由于我们只需要实现一个框架，所以我们只需要关心 WSGI 的框架端，我们不需要实现服务器端的接口。RestPy3 不提供服务器的实现，但是它可以在任何实现了服务器网关接口服务器端的 Web 服务器上运行，默认情况下，RestPy3 框架运行在 Python 自带的测试服务器上。如果我们需要指定使用其它服务器，只需要改变 run 方法中的 make_server 参数即可。

第三章　需求建模

3.1　需求概述

RestPy3 主要是为了提供一个快速、高效的创建一款 REST 风格的应用的通用框架。

1) 标记 Web 服务

   把代码中需要作为 Web 服务的类和函数，通过 API 标记出来。

2) 建立 Web 服务集合

   从被标记的 Web 服务中，建立 Web 服务集合。

3) 分发 Web 服务请求

   通过 URL 映射，找到指定 Web 服务进行响应。

4) 响应 Web 服务请求

   根据请求，返回相应的资源。

3.2　用例图

结合需求概述，如图 3-1 为用例图：

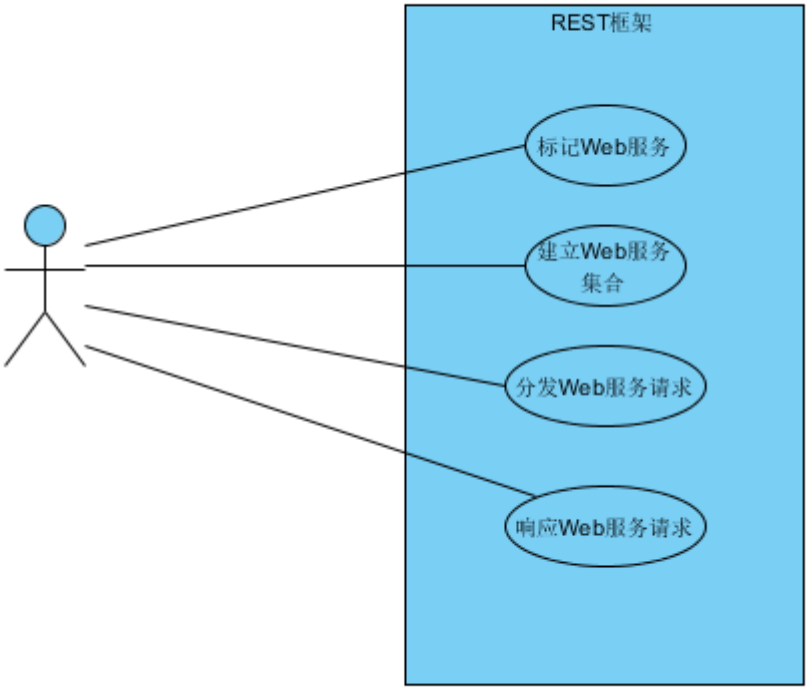

图 3-1：用例图

3.2.1 标记 Web 服务

用户可以根据自己的需要，标记出 Web 服务。如下所示：

用例 UC1：标记 Web 服务

范围：框架系统 级别：子功能 主要参与者：RestPy3 框架 前置条件：user 编写好提供服务的类和相应的方法 后置条件：Web 服务被标记完毕 基本流程： 1. 框架提供的 API 去标记 Web 服务的路径 2. 框架提供的 API 去标记 Web 服务接受的方法 3. 框架提供的 API 去标记 Web 服务接受的资源格式 4. 框架提供的 API 去标记 Web 服务返回的资源格式 备选流程： 1. 标记使用错误，导致框架抛出异常

3.2.2 建立 Web 服务集合

系统运行系统后，系统根据 API 标记，建立起 Web 服务集合。如下所示：

用例 UC2：建立 Web 服务集合

范围：框架系统
级别：子功能
主要参与者：RestPy3 框架
前置条件：user 标记 Web 服务被完毕，且系统已经启动
后置条件：Web 服务集合建立完毕
基本流程：
1. 系统运行
2. 系统收集被标记的类和方法
3. 根据标记，构造一个 Web 服务，并初始化 Web 服务的各项参数
4. 把服务加入 Web 服务集合，直到所有 Web 服务都加入到 Web 服务合集为止
备选流程：
1. 没有被标记的类和函数，Web 服务集合为空

3.2.3 分发 Web 服务请求

当系统接收到一个 Web 服务请求时，系统通过 URL 映射，选择正确的 Web 服务。如下所示：

用例 UC3：分发 Web 服务请求

范围：框架系统
级别：子功能
主要参与者：RestPy3 框架
前置条件：系统运行，且服务集合初始化完毕
后置条件：找到与 Web 服务请求对应的 Web 服务
基本流程：
1. 系统接收到一次 Web 服务请求
2. URL 映射根据 Web 服务请求，找到相应的 Web 服务
3. 返回相应 Web 服务
备选流程：
1. 没有找到与 Web 服务请求对应的 Web 服务

3.2.4 响应 Web 服务请求

当系统接收到一个 Web 服务请求时，系统通过 URL 映射，选择正确的 Web 服务。如下所示：

用例 UC4：响应 Web 服务请求

范围：框架系统 级别：子功能 主要参与者：RestPy3 框架 前置条件：系统接收到一次 Web, 并且找到与 Web 服务请求对应的 Web 服务 后置条件：系统响应 Web 服务请求 基本流程： 1. 系统找到与 Web 服务请求对应的 Web 服务 2. 根据 Web 服务请求的参数，初始化 Web 服务 3. Web 服务返回请求的资源内容， 备选流程： 1. Web 服务所需参数错误，返回错误代码

第四章　架构设计

RestPy3 是一个轻量级的 REST 框架，它采用 MVC 的软件设计模式，即模型 M，视图 V 和控制器 C，通过它可以快速的建立一款 REST 风格的 Web 应用。

本章介绍了 RestPy3 所使用的设计理念，给出了各个模块之间的相互关系。

4.1　MVC 架构及原理

MVC 模式最初由 Trygve Reenskaug 在 1974 年提出，之后被施乐帕罗奥多研究中心（Xerox PARC）作为软件设计模式应用到 Smalltalk 上。MVC 模式的目的是为了实现动态的软件设计，使从而使后续对软件的修改和扩展变得简单，并且使程序某一部分的重复利用成为可能。除此之外，此模式通过对复杂度的简化，使程序结构更加直观。软件系统通过对自身基本部份分离的同时也赋予了各个基本部分应有的功能。专业人员可以通过自身的专长分组。[18]

4.1.1 MVC 定义

MVC 模式（Model-View-Controller）是软件工程中的一种软件架构模式，它把软件系统分为三个基本部分：模型（Model）、视图（View）和控制器（Controller）。这种分离关注点的方法使得各个模块之间可以实现独立的开发、测试和维护。

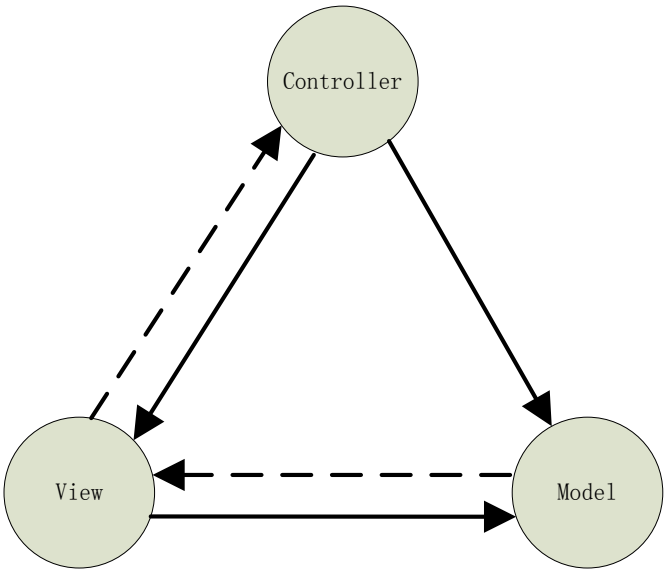

图 4-1：MVC 示意图

如图 4-1 所示，MVC 由 Model、View 和 Controller 组成：

模型（Model），数据模型用于封装与应用程序的业务逻辑相关的数据以及对数据的处理方法。“模型”有对数据直接访问的权力，例如对数据库的访问。“模型”不依赖“视图”和“控制器”，也就是说，模型不关心它会被如何显示或是如何被操作。但是“模型”中数据的变化一般会通过一种刷新机制被公布。为了实现这种机制，那些用于监视“模型”的“视图”必须事先在此“模型”上注册，从而，“视图”可以了解在数据“模型”上发生的改变。在事件驱动的系统中，当“模型”中的数据变化时，模型会通知“视图”，从而使“视图”响应变化。

视图（View），视图层能够实现数据有目的的显示，理论上视图层不是必须的。在“视图”中一般没有程序上的逻辑。为了实现“视图”上的刷新功能，视图需要访问它监视的“模型”，因此应该事先在被它监视的“模型”那里注册。

控制器（Controller），“控制器”起到不同层面间的组织作用，用于控制应用程序的流程。它处理事件并做出响应。“事件”包括用户的行为和数据模型上的改变。

4.1.2 MVC 的优缺点

MVC 的优点体现在以下几个方面：

1. 多个视图可以共享一个模型。在 MVC 模式中，模型响应用户请求并返回相应的数据，视图负责格式化数据并把最终把数据呈现出来。这样，就达到了业务逻辑层和表示层的分离。很多情况下，业务逻辑是单一的，而显示方式是不同的。例如，同一套业务逻辑，可能既需要通过网页的方式呈现，也需要通过客户端的方式展现给用户。这时，同一个模型可以被不同的视图重用，这样就大大提高了代码的可重用性。而且，由于模型返回的数据不带任何显示格式，因而这些模型也可直接应用于其它接口，不仅仅限于视图的使用。
2. 一个应用被分离为三个不同的部分，很多情况下改变其中的一层就能满足整个应用的改变。比如一个应用从 C/S 架构改变到到 B/S 架构，可能只需要替换视图这部分。而应用的业务流程或者业务规则的改变则只需改动模型这部分。控制器把不同的模型和不同的视图组合在一起响应各类的请求，因此，控制器可以说是包含了用户请求权限的概念。由于不同的部分各自职责分明，同一部分不同的应用具有某些相同的特征，从而也有利于通过工程化、工具化生成和管理程序代码。

3. 控制器是自包含（self-contained），即高内聚低耦合的部分，与模型和视图保持相对独立，所以可以方便的改变应用程序的数据层和业务规则。例如，把数据库从 MySQL 移植到 Oracle，或者把 RDBMS 数据源改变成 LDAP 数据源，只需改变控制器即可。一旦正确地实现了控制器，不管数据来自数据库还是 LDAP 服务器，视图都会正确地显示它们。由于 MVC 模式的三个模块相互独立，改变其中一个不会影响其他两个，所以依据这种设计思想能构造良好的少互扰性的构件。此外，控制器提高了应用程序的灵活性和可配置性。控制器可以用来连接不同的模型和视图去完成用户的需求，也可以构造应用程序提供强有力的手段。给定一些可重用的模型和视图，控制器可以根据用户的需求选择适当的模型机型处理，然后选择适当的的视图将处理结果显示给用户。

MVC 的不足体现在以下几个方面：

1. 增加了系统结构和实现的复杂性。对于简单的系统，严格遵循 MVC，使模型、视图与控制器分离，会增加结构的复杂性，并可能产生过多的更新操作，降低运行效率。
2. 视图与控制器之间的紧耦合。在 MVC 中，视图与控制器虽然是相互分离，但在很多情况下，确实联系紧密的部件，视图在没有控制器的存在，其应用价值是很有限的，反之亦然，这样所谓的独立重用就显得不切实际。
3. 视图对数据模型的访问效率低。依据模型操作接口的不同，视图可能需要多次调用才能获得足够的显示数据，从而使运行效率降低。而对未变化数据的不必要的频繁访问，也将降低系统的性能。
4. 由于 MVC 没有明确的定义，所以完全理解 MVC 模式并不是很容易。使用 MVC 模式需要精心的计划，由于它的内部原理比较复杂，所以需要花费一些时间去思考。
5. 由于 MVC 模式将一个应用程序分成了三个部件，所以同一个工程将比非 MVC 模式的应用包含更多的文件。

在最初的 JSP 网页中，像数据库查询语句这样的数据层代码和像 HTML 这样的表示层代码混在一起。经验比较丰富的开发者会将数据从表示层分离开来，但这通常不是很容易做到的，它需要精心地计划和不断的尝试。MVC 从根本上强制性地将它们分开。尽管构造 MVC 应用程序需要一些额外的工作，但是它带给我们的好处是毋庸

置疑的。

过去 MVC 模式并不适合小型甚至中等规模的应用程序，这样会带来额外的工作量，增加应用的复杂性。但现在多数软件设计框架，能直接快速提供 MVC 骨架，供中小型应用程序开发，此问题不再存在。对于开发存在大量用户界面，并且逻辑复杂的大型应用程序，MVC 将会使软件在健壮性、代码重用和结构方面上一个新的台阶。尽管在最初构建 MVC 模式框架时会花费一定的工作量，但从长远的角度来看，它会大大提高后期软件开发的效率。

使用 Python 语言实现的 Web 框架有很多，比如 Django、Enthought、Pylons、PureMVC 等。

4.2 领域模型

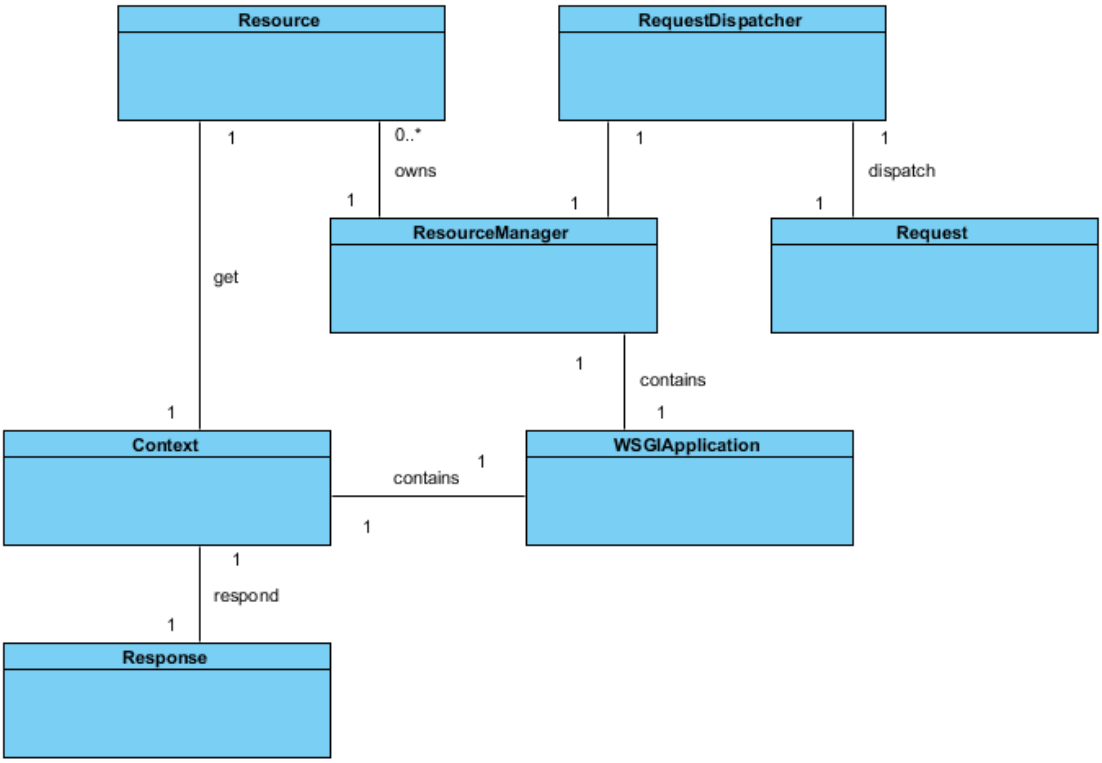

图 4-2：领域模型图

如图 4-2 所示，程序运行时，WSGIApplication、ResourceManager、Context 会初始化完毕。

当一次请求（Request）到来时，由 RequestDispatcher 向 ResourceManager 发出请求，从而获取相应的资源（Resource），程序最后做出响应（Response）。

第五章　模块设计

本章介绍了 RestPy3 框架各个模块的作用，同时也着重介绍了 RestPy3 框架中比较重要一些类。

RestPy3 框架由 Python 3 编写而成，设计与实现也遵循 Python 的编码规范。

5.1　core 模块

core 模块包括框架对外提供的用于标记资源的 API，如表 5-1、5-2、5-3 所示：

表 5-1：标记 HTTP 方法 API 表

API	描述	示例
get	若要检索某个资源，应该使用 GET 方法。	GET /adduser?name=Robert HTTP/1.1
post	若要在服务器上创建资源，应该使用 POST 方法	POST /users HTTP/1.1
delete	若要删除某个资源，应该使用 DELETE 方法	DELETE /users/Robert HTTP/1.1
put	若要更改资源状态或对其进行更新，应该使用 PUT 方法。	PUT /users/Robert HTTP/1.1

表 5-2：标记资源格式 API 表

API	描述	备注
consume	用于标记 Web 服务所接受的信息格式	无
produce	用于标记 Web 服务所返回的信息格式	目前只支持 JSON 格式

表 5-3：标记路径 API 表

API	描述	备注
path	用于标记 Web 服务的资源路径	无

5.2 application 模块

application 模块中包含了与框架相关的函数、类、以及变量，如图 5-1 所示：

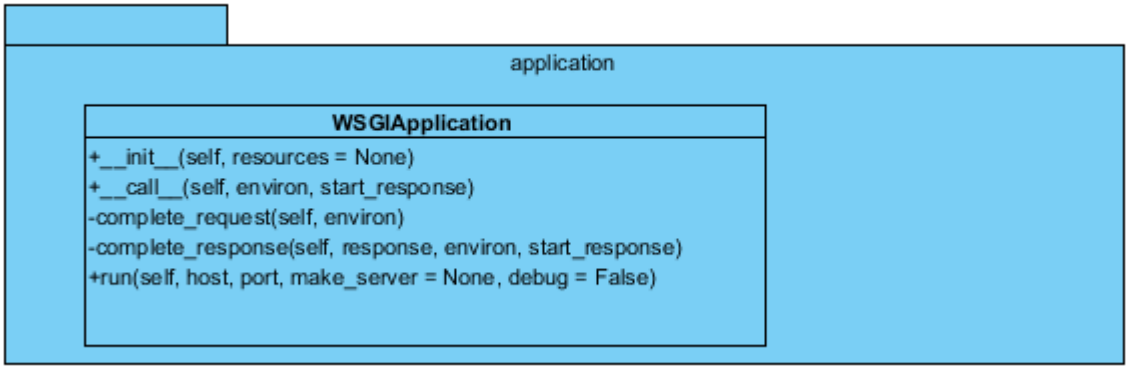

图 5-1：application 模块包图

WSGIApplication 类主要实现了服务网关接口（WSGI）的框架端接口，代表框架程序，框架需要通过 run 方法来启动。WSGIApplication 类通过__call__(self, environ, start_response)方法实现接口，默认情况下，WSGIApplication 使用 Python 自带的测试 Web 服务器。

5.3 resource 模块

resource 模块中包含了与资源相关的函数、类、以及变量，如图 5-2 所示：

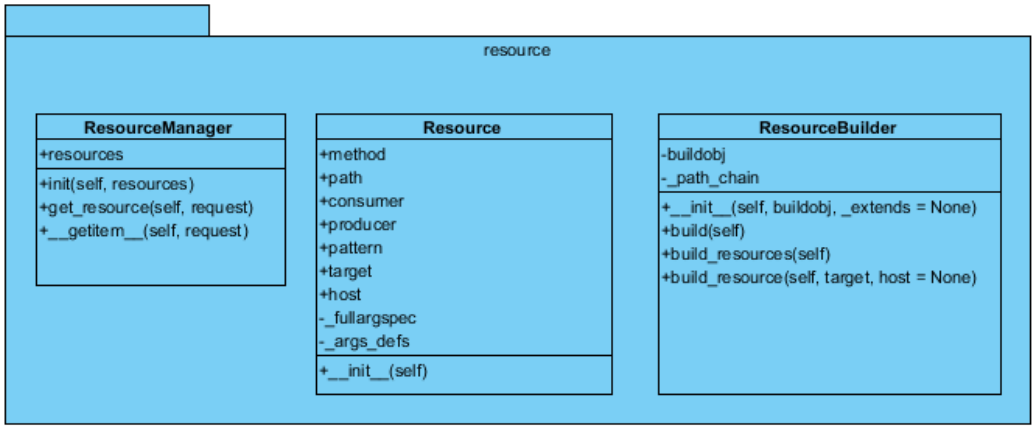

图 5-2：resource 模块包图

Resource 类代表了 Web 应用程序中的资源，用于表示对外提供的 Web 服务。ResourceBuilder 类负责抽取被 core 模块中 API 标记的资源的信息，并构建出

Resource 类。

ResourceManager 类负责管理所有的可用的 Resource 类。get_resource 类通过接受 request 类，从而返回一个响应的 resource 类，相当于通过判断不同的 request 类，通过 URL 映射系统，从而选择一个 resource 类与之对应。

5.4 context 模块

context 模块中包含了管理程序运行的当前的上下文环境所要用到函数和类，如图 5-3 所示：

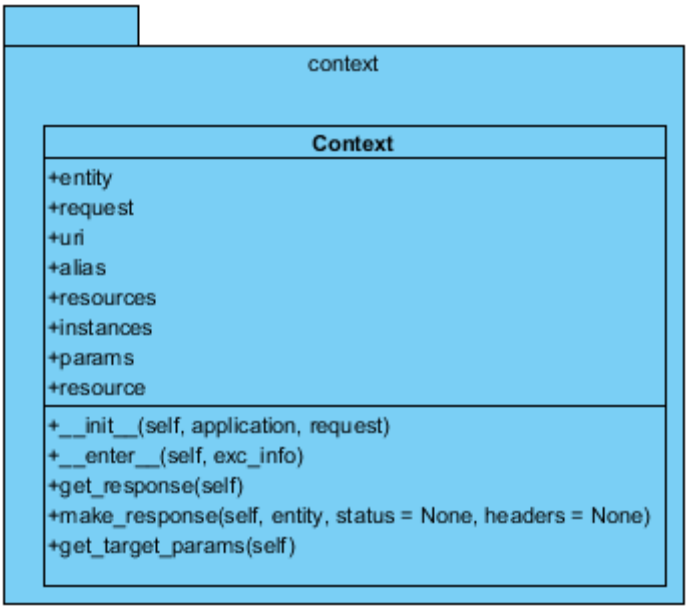

图 5-3：context 模块包图

Context 类负责管理程序运行的当前的上下文环境，例如，保存会被用到的 resource 类，并通过 make_response 方法来返回一个 response 类。

5.5 request 模块

request 模块中包含了与请求相关的函数、类、以及变量，如图 5-4 所示：

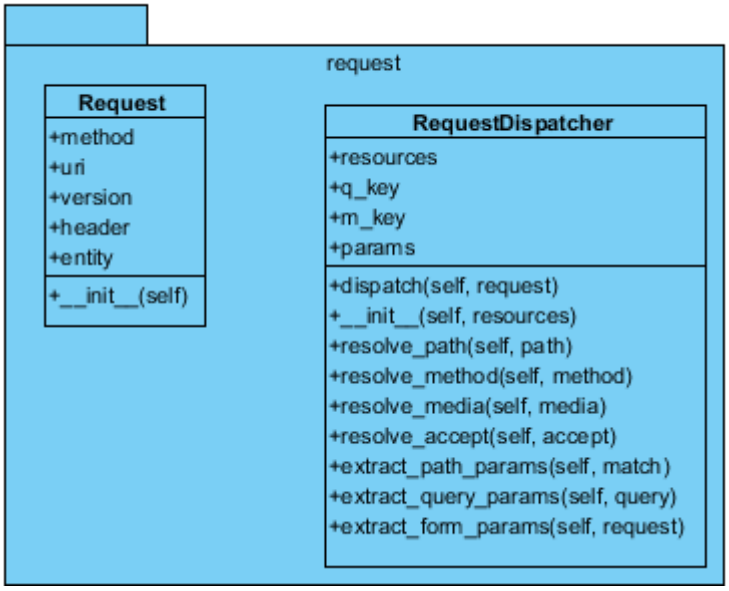

图 5-4：request 模块包图

Request 类代表对程序进行的一次资源请求，会包含请求的方法、请求的资源的路径，请求所带的参数等。

RequestDispatcher 类负责把一次资源请求分发到具体的 resource 类上进行处理，从而组成一个简单的 URL 映射系统。

5.6 response 模块

response 模块中包含了与响应相关的函数、类、以及变量，如图 5-5 所示：

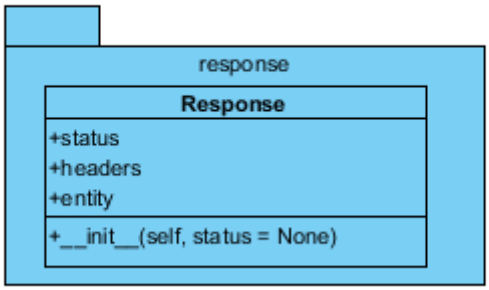

图 5-5：response 模块包图

Response 类代表程序对一次请求的响应，status 代表返回状态，headers 为返回的头部信息，entity 为返回的信息的实体。

第六章　部署与应用

本章介绍了 RestPy3 的运行环境和部署方法，并简单的介绍了如何使用 RestPy3 快速的编写一款 REST 风格的 Web 应用。

6.1　运行环境

由于本框架只依赖于 Python 标准库，所以唯一的使用前提就是安装 Python 3。

6.2　快速搭建一个应用

使用 RestPy3 编写一个 REST 风格的应用是非常简单的。步骤如下：

1. 安装 Python 3。
2. 在编写的 Web 应用程序中引用本框架。
3. 对需要提供的 Web 服务进行标记。

简单的 hello world 程序如下：

```
import rs
@rs.path('/hello')
@rs.get
def helloservice(self):
    return 'hello world!'
application = rs.application()
application.run('', 8001, debug=False)
```

运行程序后，访问http://localhost:8001/hello, 即可得到“hello world!”输出。

可以看出，RestPy3 框架使用方便简单，通过对代码进行标记，就可以构建 REST 风格的 Web 应用。

第七章　总结与展望

随着 REST 的普及和深入，随着 ROA 将扮演这越来越重要的角色。一个好的 REST 风格的框架，将可以大大节约开发时间。

7.1　总结

本文从设计到实现，介绍了如何构建一个 REST 框架，并给出了一个参考实现 RestPy3。

通过 RestPy3 框架可以快速、高效的创建一款 REST 风格的应用。RestPy3 框架方便易用，拥有和 Jersey 类似的 API，使用过 Jersey 的用户可以很快的学会如何使用。

而且，目前支持 Python 3 的框架十分少，现有的大部分框架都基于 Python 2 编写，很多框架由于代码量大、依赖的类库比较多，所以一时间很难快速去支持 Python 3。RestPy3 由于是完全基于 Python 3 编写，所以在选择 Python 3 作为开发语言的人群中，还是有一定优势的。

7.2　展望

本文中的 RestPy3 框架只是一个简单的实现，并不具有在真是生产环境中使用的能力，还有很多地方有待改进，如：

1. RestPy3 框架没有缓存机制。今后将会增加完善缓存机制。
2. RestPy3 框架缺少有效地异常处理机制，在真正的生产环境中，异常处理十分重要。今后将会增加完善的异常处理机制。

RestPy3 框架与现有的框架，如 Web.py、Django 相比还有很大的差距，随着对 Web 框架的深入学习，将对其持续改进。

致谢

光阴似箭，一转眼大学生活就已经接近尾声了。

在这四年里中，我很高兴认识了这么多可爱的大学同学，因为有了你们，我的大学生活才如此的丰富多彩；感谢那些参加比赛时，和我一起奋斗过的同学们，因为你们，我的才明白了什么是团队合作；感谢那些参加社团时，和我一起忙碌过的同学和老师，因为你们，我们才使我们的社团走向更好。

感谢赵淦森老师的指导，我之所以会选择安全方向作为未来研究生的方向，也是受到赵淦森老师的影响。

感谢那些在我学业上悉心指导我的老师，因为有了你们，我的大学生活才如此的充实，并沿着自己喜欢的方向去奋斗。

感谢中山大学，感谢软件学院，在这里，我学到了很多，这些知识将伴随我的一生，引导我向着自己的梦想去奋斗。

最后，再次感谢我身边的每一个人，感谢四年来的帮助，谢谢大家！

参考文献

[1] Ross Mason, How REST replaced SOAP on the Web: What it means to you. http://www.infoq.com/articles/rest-soap

[2] Alex Rodriguez, RESTful Web services: The basics. http://www.ibm.com/developerworks/cn/webservices/ws-restful/

[3] Fielding, Roy T.; Taylor, Richard N. (2002-05), "Principled Design of the Modern Web Architecture" , ACM Transactions on Internet Technology (TOIT) (New York: Association for Computing Machinery) 2 (2): 115–150, doi:10.1145/514183.514185, ISSN 1533-5399

[4] Fielding, Roy Thomas (2000), Architectural Styles and the Design of Network-based Software Architectures, Doctoral dissertation, University of California, Irvine

[5] Pautasso, Cesare; Zimmermann, Olaf; Leymann, Frank (2008-04), "RESTful Web Services vs. Big Web Services: Making the Right Architectural Decision", 17th International World Wide Web Conference (WWW2008) (Beijing, China)

[6] Richardson, Leonard; Ruby, Sam (2007-05), RESTful Web Services, O'Reilly, ISBN 978-0-596-52926-0

[7] Richardson, Leonard; Ruby, Sam, RESTful Web Services, O'Reilly. 2007, ISBN 0596529260

[8] SOAP Version 1.2 Part 1: Messaging Framework (Second Edition), http://www.w3.org/TR/soap12-part1/

[9] 李三红，Web 服务编程，REST 与 SOAP。 http://www.ibm.com/developerworks/cn/webservices/0907_rest_soap/

[10] Elkstein, M. What is REST? http://rest.elkstein.org/2008/02/what-is-rest.html

[11] http://www.json.org/json-zh.html

[12] 廖雪峰，JSON 入门指南，

http://www.ibm.com/developerworks/cn/web/wa-lo-json/

[13] http://zh.wikipedia.org/wiki/JSON

[14] "XML Media Types, RFC 3023". IETF. http://tools.ietf.org/html/rfc3023

[15] M. Murata, D. Kohn, and C. Lilley (2009-09-24). "Internet Drafts: XML Media Types". IETF. http://tools.ietf.org/html/draft-murata-kohn-lilley-xml-03

[16] http://www.regular-expressions.info/

[17] http://www.python.org/dev/peps/pep-3333/

[18] http://en.wikipedia.org/wiki/Model-view-controller

毕业论文成绩评定记录

指导教师评语：

该论文是一篇工程论文，主要论述了 REST 框架的设计与实现，主要工作是考察现有的 REST 框架，并给出自己的设计和改进。该论文按软件工程的规范详细描述了从需求分析、框架设计到编写实现系统的完整流程，并在 Windows 和 Linux 系统下进行了测试。论文最后给出了使用 Python 编写的一个参考实现，运行结果比较符合要求。该论文对于 REST 框架的设计与实现有一定的参考价值，但在创新性上有所欠缺。论文的格式、内容的组织与写作基本符合规范。论文作者已具备运用软件工程知识解决实际工程问题的能力。综上，本论文已达到本科生毕业论文的要求，成绩良好。

成绩评定：良

指导教师签名：　　赵淦森　　2012 年　5 月　9 日

答辩小组或专业负责人意见：

成绩评定：

签名（章）：　　年　　月　　日

院系负责人意见：

成绩评定：

签名（章）：　　年　　月　　日

附表一、毕业论文开题报告

论文（设计）题目：基于 PERMIS 模型的权限控制系统

（简述选题的目的、思路、方法、相关支持条件及进度安排等）

分布式网格环境是当前一个越来越常见的应用环境，在这种分布式网格环境下关于用户权限控制的问题是一个难以解决的，有许多现存问题的问题，本论文即对此提出一种设计及实现方案，最终这样一个方案也可以应用到云环境下。

针对这个问题，我计划参考 PERMIS 模型来进行设计。该系统是基于 Policy 的，根据一系列之前人为定义的存储在 CIS 或者 AR 中的 Policy 来控制模块中权限控制的运作，使系统更加灵活可扩展。只要去修改相应的 Policy 就可以修改和更新权限控制。此外也可以实现控制一个领域的权限分发而允许用户之间相互授予权限。最为关键的是可以实现每块资源可以单独控制这些权限的验证，并且允许每块资源域独立的去信任那些分发权限（credentials issueing）的机构，并独立决定验证申请需要包含哪些内容，这是当今很多 Grid System 中所不含有的。我打算突出该模型的这些特征，建立一个较为完善的权限控制系统。

我计划的进度安排：

阶段 1，开题至本学期末：深入学习 PERMIS 系统，分析其各个模块间的作用，完成工程阶段所需的知识储备学习，总结一份学习总结报告，完成论文的框架。

阶段 2, 截止 3 月 10 日，这个阶段我针对第一阶段的所得，完成项目的大体功能。

阶段 3，截止 4 月 10 日，通过测试完成项目并进行实验验证其各个特征，完成毕业设计报告交给老师审核。

学生签名：　　高浩　　2011 年 11 月 29 日

指导教师意见：
同意开题

1、同意开题（ √ ） 2、修改后开题（ ） 3、重新开题（ ）

指导教师签名： 赵淦森 2011 年 12 月 1 日

附表二、毕业论文过程检查情况记录表

指导教师分阶段检查论文的进展情况（要求过程检查记录不少于 3 次）：

第 1 次检查

学生总结：

通过阅读相关文献和资料，对毕业设计也有了初步构思。 确定选择基于PERMIS 模型的权限控制系统作为自己的题目。

指导教师意见：

同意开题。

第 2 次检查

学生总结：

本阶段放弃了之前的开题，重新选择题目，REST框架的设计与实现。

指导教师意见：

同意更换题目。

第 3 次检查

学生总结：

通过阅读资料，掌握技术原理，完成了 REST 框架的需求分析、架构设计等。

指导教师意见：

进度需要加快。

第 4 次检查

学生总结：

完成了 REST 框架的实现。

指导教师意见：

基本符合要求，拟评为良。

学生签名：　　高浩　　2012 年 5 月 9 日

指导教师签名：　　赵淦森　　2012 年 5 月 9 日

总体完成情况	指导教师意见： 1、按计划完成，完成情况优（ ） 2、按计划完成，完成情况良（ √ ） 3、基本按计划完成，完成情况合格（ ） 4、完成情况不合格（ ） 指导教师签名：　　赵淦森　　2012 年 5 月 9 日

附表三、毕业论文答辩情况

答辩人		专 业	
论文（设计）题目			
答辩小组成员			

答辩记录：

记录人签名：　　年　　月　　日

中山大学软件学院

本科生毕业论文形式自查审查表

学号	08386145	姓名	高浩	专业方向	软件工程
论文题目	中文	REST 框架的设计与实现			
	英文	Design and Implementation of a REST framework			

论文是否有以下部分		格式审查	
1、中文摘要	√	1、是否列出至少 10 篇以上的参考文献	√
2、中文关键词	√	2、参考文献格式是否正确	√
3、英文摘要	√	3、所列参考文献是否被引用	√
4、英文关键词	√	4、英文术语或缩写的大小写是否规范	√
5、目录	√	5．目录、章节编号是否规范	√
6、致谢	√	6、图、表、公式的编号是否正确	√
7、参考文献	√	7、图、表是否有标题，并在正文中正确引用	√
8、正文页码	√	8、正文篇幅是否足够（不少于 1 万字。注：正文内容，不包括图表、参考文献等）	√

形式自查学生签名	高浩	形式自查时间	2012/4/23
二次形式审查意见		二次形式审查时间	

注：在相应审查项右侧打勾表示通过该项审查；打叉表示不通过该项审查。

学术诚信声明

本人所呈交的毕业论文，是在导师的指导下，独立进行研究工作所取得的成果，所有数据、图片资料均真实可靠。除文中已经注明引用的内容外，本论文不包含任何其他人或集体已经发表或撰写过的作品或成果。对本论文的研究作出重要贡献的个人和集体，均已在文中以明确的方式标明。本毕业论文的知识产权归属于培养单位。本人完全意识到本声明的法律结果由本人承担。

本人签名：　　高浩　　日期：　　2012 年 5 月 9 日