面向资源的 RESTful Web应用研究

潘 冰

(暨南大学珠海学院 广东 519070)

摘要:作为 Web应用技术的探索与实践,提出了面向资源的 RESTful Web应用设计思路。通过对 Rails框架下 RESTful Web的实现原理进行分析,从资源的规划、控制器的设计、模型的建立、表间关联以及 RESTful URI的设计等方面对 RESTful Web应用进行了研究,给出了 Rails框架下 RESTful Web应用的开发步骤。最后,通过一个实例实现了面向资源的 RESTful Web应用。这种基于资源的设计将所有功能转化为资源,完全打破了传统的基于动作的设计思路,同时说明了 REST思想从 Web服务到 Web应用是可行的。

关键词:面向资源 Rails框架 表述性状态迁移 Web应用

Study on Resource-Oriented RESTful Web Application

PAN Bing

(Zhuhai College, Jinan University, Guangdong, 519070, China)

Abstract: To explore and practise the technologies of Web applications, the idea of resource-oriented RESTful Web application is proposed. After analyzing the implementation principles of RESTful Web based on Rails framework, the resource planning, controller design, model build, relationships between tables, as well as RESTful URI are studied, and then the steps of RESTful Web application development on Rails are provided. Finally, an example of resource-oriented RESTful Web application is implemented. This resource-based design transforms all functions into resources, completely breaks the traditional action-based design ideas, and the example shows that the REST ideas from the Web services to Web applications is feasible.

Keywords: Resource-oriented, Rails framework, REpresentational State Transfer (REST), Web application

传统的 Web应用程序的设计主要是基于动作(action)的设计思想,并且有多种框架支持这种思想的实现。2000年 Roy Thomas Fielding在他的博士论文[1]中提出了 REST(REpresentational State Transfer,表述性状态转移)术语,它作为分布式超媒体系统设计的一种架构风格,将资源和资源的表示分开,为构建高性能、简单性、可移植性的 Web程序提供了一个架构风格上的准则[2]。在 REST中,整个 Web被看作一组资源的集合,资源是一个基于 URI(Universal Resource Identifier)的实体,包括实体对象和业务逻辑,对资源进行的操作由客户端指定的 URI和 HTTP协议动词的组合来实施的。REST这种基于资源的设计改变了传统的基于动作的设计思想。

目前,基于 RESTful(简称 REST)风格[3]的 Web Services在一些领域得到较好了的应用。一些大公司已经提供了基于 REST的网络服务,如 Google服务(如 http://code.google.com/intl/zh-CN/)、Amazon S3、Yahoo!提供的大部分 Web服务(http://developer.yahoo.com),提供类似 Web浏览器书签服务的网站 del.icio.us采用 REST风格,但并非一个真正的 RESTful Web应用[3]。虽然静态网站具有典型的 REST风格,但是建立在网站之上的能够为用户提供动态服务的 RESTful Web应用程序在国内还不多见。本文在对基于资

本文于 2010-04-16收到,2010-06-12收到修改稿。

源的架构和 RESTful Web研究基础上,探讨利用 Rails框架实现 RESTful Web应用开发过程,并以实例展示基于资源的 Web开发思想和方法。

1 REST

REST以资源为中心,任何事物,只要具有被引用的必要,它就是一个资源。在 REST的世界里,整个 Web被看作一组资源的集合。每个资源必须至少有一个统一资源标识符 URI,URI既是资源的名称,也是资源的地址。REST中的资源是数据和表现形式的组合,以资源为核心的设计思想是 REST的核心所在。REST指的是一组架构规范和原则。REST规范包括客户/服务器,无状态,缓存,统一接口、分层系统和按需代码,REST针对 Web应用提出了如下设计准则[4]:

(1)网络上的所有事物都被抽象为资源(resource);

(2)每个资源对应一个唯一的资源标识符(resource identifer);

(3)通过通用的连接接口对资源的操作;

(4)对资源的各种操作不会改变资源标识符;

(5)所有的操作都是无状态的(stateless)。

满足 REST规范和原则的应用程序或设计就是 RESTful[5]。RESTful Web应用系统是一个面向资源的系统,具有如下特点[6]:

(1)简明的 URI

REST URI代表资源,其结构由控制器的名称和资源的 id组成,并不包含对资源的动作。

(2)统一接口

REST要求使用统一的接口,统一接口独立于资源的 URI。任何对资源的操作行为都是通过 HTTP协议来实现:获取资源的一个表示用 HTTP GET;向一个新的 URI发送 HTTP PUT或向一个已有 URI发送 HTTP POST创建一个新资源;修改已有资源用 HTTP PUT;删除已有资源用 HTTP DELETE。

(3)资源可以有多种表示方法

对于控制器的同一个 action,可以返回给客户端 html、xml或 RSS不同格式的内容,依赖于客户端的调用方式。

(4)面向 CRUD的控制器

所谓 CRUD就是 Create、Read、Update和 Delete的简写[7]。每个控制器都对应某个资源的 CRUD操作。

(5)更少的代码和简单的系统设计。

2 Rails框架

Rails框架是 Ruby on Rails框架的简称,也称 Rails,是一个基于 Ruby语言的网络应用软件开发框架。Rails1.2起开始提供了关于 REST的支持,让 REST的思想从 Web服务的应用提升到了 Web应用软件开发。

Rails基于"Model—View—Controller"(MVC),主要包括三个库:ActiveRecord库实现 MVC中的 Model,它封装数据库表或视图中数据行的对象,封装数据库访问过程;ActionView库实现 MVC中的 View,它是一个基于嵌入式 Ruby(ERb)的系统,用于定义服务与数据表示的表示模板;ActionController库实现 MVC中的 Controller,处理和组织来自数据库和 Web表单输入的数据,然后它将这些数据提交给 ActionView,ActionView将这些数据插入模板并显示。在 MVC中,V和 C关系非常紧密,所以 ActionView和 ActionController捆绑在一起成为 ActionPack[8]。

3 Rails实现 RESTful Web应用的原理

目前,Rails2.3.4已全面基于 REST,Rails框架不允许直接定义资源,它把 Web应用的功能划分为一个

个控制器,由控制器来暴露资源[3],一个控制器属于某一个特定的资源。Rails控制器具有 CRUD行为功能,由一些标准动作来执行某个资源的 CRUD操作。通过控制器的设计,明确应该暴露哪些资源,响应哪些 URIs以及统一接口方法。

Rails的 routes.rb文件通过 map.resources来定义 RESTful URI,它是基于资源的路由。RESTful URI由控制器和模型类组成,URI中不含动作,通过 HTTP的 4个动词和 REST URI的组合形成对某个资源的操作。Routes.rb负责将用户的请求路由到指定的方法,通过控制器类告诉 Rails如何把收到的请求路由给哪个类,以及如何对给定请求进行处理。当 Rails收到一个 HTTP请求时,它会根据该请求的目标 URI的第一个路径变量把该请求分配给适当的控制器类处理。例如,为"questions"资源定义 RESTful URI:

```
ActionController::Routing::Routes.draw do |map|
    map.resources :questions
end
```

map.resources自动为应用程序创建 7个路由规则如表 1所示,同时将 HTTP动词映射到控制器 action。对于 QuestionsController控制器类,默认方法限制在 index、show、new、edit、create、update和 destroy,通过它们来执行对资源"questions"的 CRUD操作。

表 1 RESTful URI、HTTP动词和 Action的对应关系[7]

Path—Method	HTTP Verb	REST—URI Path	Action	操作描述
questions_path	GET	/questions	index	返回问题列表
questions_path	POST	/questions	create	提出新问题
question_path(:id)	GET	/questions/:id	show	返回 params[:id]所标识的资源
question_path(:id)	PUT	/questions/:id	update	更新 params[:id]所标识的资源
question_path(:id)	DELETE	/questions/:id	destroy	删除 params[:id]所标识的资源
new_question_path	GET	/questions/new	new	为创建新资源提供表单
edit_question_path(:id)	GET	/questions/:id/edit	edit	以 form形式返回 params[:id]标识的资源内容以便更新

由于目前流行的 Web浏览器大多不支持 PUT和 DELETE方法,所以 Rails采用一个虚拟的:method参数来模拟 HTTP中的 PUT和 DELETE方法。

在 Rails控制器中通过 respond_to负责解释客户端的请求,然后按客户端所需要的不同的信息格式返回不同格式的内容。默认格式有 HTML和 XML。

4 基于 Rails的 RESTful Web应用设计步骤

由于 Rails对 REST的支持,在设计基于 Rails的 RESTful Web应用程序时,从资源入手并对资源进行规划。所有可以被抽象为资源的东西都为其定义 RESTful URI,每个 URI都代表一个资源,而整个系统就是由这些资源组成的。通过 URI来设计系统的结构,这是 RESTful Web应用设计的关键所在。基于 Rails的 RESTful Web应用设计步骤:①规划数据集;②创建模型;③设计控制器。把数据集分配给一个个控制器,定义资源 URI,根据系统的功能来创建资源,而不是通过创建 action来实现业务逻辑。④路由。在 routes.rb中通过 map.resources配置 REST路由,对于嵌套资源,Rails也提供了在 routes.rb中增加了路由的定义。⑤设计视图。Rails视图主要通过采用 ERb模板即一种 HTML和 Ruby代码的混合来定义的。视图中用到了 routes.rb中生成的 helper方法。用户通过链接和按钮来和系统交互以及资源之间的链接和状态的转移。

5 面向资源的 RESTful网络答疑系统

作为面向资源的 RESTful web应用设计思想的实现,本文以网络课程支撑平台的一个子系统——网络

答疑系统为例介绍如何在 Rails框架下实现 RESTful Web应用开发。该系统主要通过网上答疑来解决用户的提问,用户可以提问并对问题进行维护管理,可以显示自己的提问、查询、浏览典型问题和最新问题。具体步骤如下:

(1)规划数据集

作为简化的版本,该系统涉及到的数据有用户、问题和问题答案。"用户"分为学生和教师,学生具有用户名称、密码和 Email属性;教师有姓名、密码、简介和 Email属性。"问题"指学生的提问,从属于学生,包括学生 id问题标题、内容和是否解答属性;"问题答案"从属于"问题",同时从属于教师,包括教师 id问题 id内容、回答时间和是否典型问题属性。

(2)创建模型

基于本实例中的实体,建立 4个数据表 students、teachers、questions和 answers。对应的 4个模型文件以及表间关系如下:

```
class Student< AetiveReeord::Base
  has_many:questions  #students与 questions之间 1对多关系
end
class Teacher< ActiveRecord::Base
  has_many:answers  #teacher与 answers之间 1对多关系
end
class Question< ActiveRecord::Base
  belongs_to:student  #指明从属关系
  has_one:answer  #questions与 answers之间 1对 1关系
end
class Answer< ActiveRecord::Base
  belongs_to:teacher  #指明从属关系
  belongs_to:question  #指明从属关系
end
```

(3)设计控制器

采用基于资源的设计思想需要把应用的功能划分为一个个控制器,由控制器来暴露资源。因此,控制器的设计就是资源的设计,系统需要实现的功能是对资源的定义及命名,因此资源的命名最好名词化。Rails控制器用 Ruby类来实现。本系统设计了 8个控制器。

学生控制器 students_controller实现对学生资源的 CRUD操作,该控制器(类)的根 URI为 /students。

教师控制器 teachers_controller实现对教师资源的 CURD操作,该控制器的根 URI为 /teachers。

问题控制器 questions_controller从属于学生资源,该控制器的根 URI为 /students/{id}/questions。"我的问题"由 QuestionsController#show方法完成,对应 URI为 /students/{id}/questions。"我要提问"由 QuestionsController#new方法完成,对应 URI为 /students/{id}/questions/new。

对于非 CRUD操作,如"最新问题"、"典型问题"和"问题查询"等功能,将其资源化后对应的控制器、URI、HTTP动词、action以及操作描述如表 2所示。

表 2　资源化后的控制器、RESTful URI及 action

控制器	HTTP动词	REST URI	Action	操作描述
recent_questions_controller	GET	/recent_questions	index	显示最新问题
typical_questions_controller	GET	/typical_questions	index	显示典型问题
search_questions_controller	GET	/search_questions	index	提供表单页面,供用户输入关键字
search_questions_controller	POST	/search_questions	create	执行查询

答案控制器 answers_controller从属于问题资源。该控制器的根 URI为 /questions/{id}/answers。教师"解答问题"实际上是创建资源,由 AnswersController#new和 AnswersController#create来对答案表的 create操作。

最后是用户登录和退出控制器 login_states_controller,根 URI为 /login_States。登录和退出对应的 Rails方法分别为 LoginStatesController#create和 LoginStatesController#destroy。

(4) routes. rb文件

资源创建好之后,需要将 HTTP动词、资源 URI和 action联系起来,即将请求路由到资源并进行处理。Routes. rb文件部分内容如下:

```
map. resources: students do | student|
  student. resources: questions  #定义 questions资源路由
end
map. resources: questions do | question|
  question. resources: answers  #answers是嵌套资源
end
map. resources: teachers do | teacher|
  teacher. resources: answers  #answers是嵌套资源
end
  map. resources: recent_questions  #定义资源路由。下同
  map. resources: typical_questions
  map. resources: search_questions
  map. resources: login_states
```

(5)设计视图

视图文件对应各控制器的 action。以“问题查询”为例简单介绍相关代码。提供页面供用户输入关键字的视图文件 index. html. erb主要内容如下所示:

<%#以下代码使用 form_tag辅助方法构造表单,提供查询所需关键字的输入。通过 params[:title]返回参数,提交后通过 SearchQuestion控制器的 create方法完成查询工作。%>

```
Search Question
<%  form_tag do%>
  <label for="title">Keyword:</label>
  <%= text_field_tag: title, params[:title] %>
  <%= submit_tag Search%>
<%  end%>
```

查询结果的视图文件 create. html. erb主要内容:

```
Search Results  About<font color=#ff0000><%=@key%></font>
<% if@questions. empty? %>
<%#判断是否有满足条件的记录返回%>
  <%  flash[:notice] = "No Questions about<font color=#ff0000>#{@key}</font>!"%><%  else%>
  <table><%@questions. each do | question|%><tr>
  <%#循环显示满足条件的记录%>
  <td><%= link_to "#{h(excerpt(question. content, "#{@key}", 8))}", question%></td></tr>
  <%#提取关键字前后的 8个字符,构成超链,通过/question/{id}显示详细内容%>
  <%  end%>
  </table>
<%  end%>
```

(6)系统的实现

通过 Rails的脚手架(scaffold)生成模型、迁移任务、控制器、routes. rb文件和视图模型,然后进行修改和扩展后即可。如,“问题查询”控制器 search_questions_controller. rb中相关代码:

```
class SearchQuestionsController< ApplicationController
  def index
  end
  def create
    @questions= Question. find(: all, conditions=>[ "title like?", "%"+params[:title]+"%")
    #通过 params[:title]获取视图中参数,并执行查询操作,结果由实例变量@questions返回
    @key= params[:title]  #获取视图传来的参数,用实例变量@key再返回给查询结果的视图
  End
end
```

6 结束语

本文从资源的角度研究了基于 Rails框架下 RESTful Web应用的原理与开发过程,以实例实现了资源的规划、Rails控制器的创建、基于资源的路由的定义以及嵌套资源的定义与操作,为系统的功能创建相应的资源,完全没有针对功能定义 action,所有的操作都是由 HTTP动词和 REST URI来决定的。文章强调了基于

资源的思想,以及资源的 HTMI表示形式,这种完全基于资源的 WEB应用开发,不仅将对象实体资源化,并且将功能也资源化。这是 REST的思想从 Web services的应用到 Web应用程序开发的一种探索和实践。同时,我们看到了 RESTful Web应用程序和传统 Web应用程序最大的不同是 HTTP方法的使用,这种基于资源的设计思想非常适合于主要提供数据服务的 Web应用的开发。

参 考 文 献

[1] Roy Thomas Fielding. Architectural Styles and the Design of Network-based Software Architectures[EB/OL]. http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm. 2000

[2] 黄翀. REST开发框架纵览[J]. 软件世界, 2007,(17): 40-41.

[3] [美] Leonard Richardson, Sam Ruby著. RESTful Web Services中文版[M]. 徐涵, 李红军, 胡伟译. 北京:电子工业出版社, 2008

[4] 陈磊. REST的真谛[J]. 软件世界, 2007(9): 38-39

[5] REST及 RESTful的实现[EB/OL]. http://tech.sina.com.cn/s/2009-09-06/00351056986.shtml, 2009.

[6] Ralf Wirdemann, Thomas Baustert. RESTful Rails Developpment[EB/OL]. http://www.b-simple.de/download/restful_rails_en.pdf. 2007.

[7] Dave Thomas, David Heinemeier Hansson. Agile Web Development with Rails, Second Edition[M]. The Pragmatic Programmer, 2005.

[8] [美] David Black著. Ruby for Rails中文版[M]. 吴畅欣, 张明生译. 北京:人民邮电出版社, 2007.

作者简介

潘冰, 男,(1965-),硕士,副教授,主要研究方向为计算机网络, Web信息系统。