

分布式应用技术基础实验

RESTful Web Service

陈伟 (cowoo@hit.edu.cn)

<http://ims.hit.edu.cn/ds>

2007.5.29

- REST Intro
- REST from the Distributed System Perspective
- Architecture of REST
- REST on Rails
- Consuming RESTful Web Service
- Tech Community
- Next Time

- REST Intro
- REST from the Distributed System Perspective
- Architecture of REST
- REST on Rails
- Consuming RESTful Web Service
- Tech Community
- Next Time

REpresentational SState TTransfer

具有表象的状态迁移

□ What is REST

- REST是web应用的一种架构设计方式

□ Characteristic of REST

- 基于HTTP的资源
- 以HTTP协议去操作
- 数据和表象分离

Is REST Winning



□ Is REST Winning? InfoQ

“Web Services based on SOAP and WSDL are “Web” in name only. In fact, they are a hostile overlay of the Web based on traditional enterprise middleware architectural styles that has fallen far short of expectations over the past decade.

□ <http://www.infoq.com/news/2007/05/is-rest-winning>

- More and more Web services tools, such as [Apache Axis2](#) and [CXF](#), have started to offer some support for the REST model. Sun [has started JSR 311](#) to standardize support for RESTful web applications on the Java platform. Ruby on Rails has supported REST [since version 1.2](#). Most recently, [Microsoft's support for REST](#) made the news.

- REST Intro
- REST from the Distributed System Perspective
- Architecture of REST
- REST on Rails
- Consuming RESTful Web Service
- Tech Community
- Next Time

□ 分布式应用系统架构的变迁

□ 基于CORBA协议的C++中间件时代

- 对Corba有了解的人感叹，和EJB2相比，Corba实在太难用了。Corba时代在1998年EJB1.0发布以后，就逐渐淡出历史舞台了。

□ 基于RMI/IIOP协议的EJB时代

- 这个时代开始于1998年，到现在基本上已经划上了句号。其实在EJB出现以前，在1996年Microsoft发布WindowsNT4.0以后，Microsoft当时也提出了自己的分布式架构，即MTS，但是MTS的光辉被随后出现的伟大的EJB技术彻底击败，此后，就拉开了Java的应用服务器时代，BEA也是在这个时代的转折点成长起来的。

□ Corba和EJB的共同点

- 通过专有的网络协议通讯
- 不能跨平台调用
- 通过分布式对象调用来实现分布式架构，换句话说就是，分布式架构是绑定在面向对象的机制上的

□ 缺陷

- 分布式对象架构的缺陷在EJB2时代被充分暴露了出来，乃至Martin Folwer在《企业应用架构模式》当中强调，分布式调用的第一原则就是不要分布式。更多关于EJB2分布式对象架构的缺陷在Rod Johnson的《J2EE without EJB》当中被剖析的更加清楚。

□ 基于SOAP协议的Web Services时代

- 这个时代始于2001年Microsoft公司推出dotnet平台，整个行业开始鼓吹Web Services。中间经历了一次低潮之后，在IBM, BEA成功的联手炒作SOA之后，再次王者归来。

□ web services有一些明显不同于Corba和EJB分布式对象架构的特征

- 通过标准SOAP协议通讯，一般走HTTP通道
- 能够跨平台调用
- 通讯格式是xml文本，而不是二进制数据格式
- 通过RPC机制来实现分布式调用，而不是通过面向对象机制实现分布式调用

□ SOAP

- SOAP协议并不依赖于HTTP。事实上SOAP协议可以走很多底层协议，例如SMTP协议，Jabber协议等等。

□ REST也是一种分布式系统的架构风格

- REST走的是HTTP协议，并且充分利用或者说极端依赖HTTP协议
Corba和EJB是采用专有的二进制协议，SOAP可以但不依赖HTTP，并且仅仅使用HTTP POST。
- REST是基于HTTP抽象资源的分布式调用，换句话说，就是分布式调用是绑定在资源的操作上面的。

□ 对比

分布式架构

协议

调用方式

Corba架构

专有二进制协议

对象的**CRUD**操作

EJB架构

专有二进制协议

对象的**CRUD**操作

Web Services

SOAP协议

RPC方式

REST

HTTP协议

对资源的**CRUD**操作

- REST Intro
- REST from the Distributed System Perspective
- Architecture of REST
- REST on Rails
- Consuming RESTful Web Service
- Tech Community
- Next Time

Architecture of REST



GET	/showuser.jsp?userId=1
GET	/deleteuser.jsp?userId=1
POST	/adduser.jsp
POST	/edituser.jsp?userId=1

传统MVC风格
URL == Command

GET	/users/show/1
POST	/users/delete/1
POST	/users/add
POST	/users/edit/1

RoR的MVC风格
URL == Command

GET	/users/1
DELETE	/users/1
POST	/users/1
PUT	/users/1

RoR的REST风格
URL == Resource

□REST的来历

- Chapter 5 of 《Architectural Styles and the Design of Network-based Software Architectures》 written by Roy Thomas Fielding
- <http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>

□提出REST的原因

- 构建基于web的大规模分布式应用

□REST产生的背景

- 1990年，Tim Berners-Lee提出HTTP协议
- 2000年，Roy提出REST，在2000年以前的web应用都是符合REST风格的
- 随着2000年互联网热潮，以及web深入企业应用开发领域，动态web被极大扩充，同时web应用产生了诸多对HTTP协议的偏离，不再符合REST的风格
- REST是向互联网web应用的本质进行回归

□ REST的设计准则

- 网络上的所有事物都被抽象为资源(**resource**)
 - resource = data + representation
- 每个资源对应一个唯一的资源标识(**resource identifier**)
 - resource identifier = URL
- 通过连接器(**generic connector interface**)对资源进行操作
 - generic connector = HTTP Protocol (GET/POST/PUT/DELETE)
- 对资源的各种操作不会改变资源标识
 - 对资源的GET操作具有幂等性，不改变资源的状态
- 所有的操作都是无状态的(**stateless**)

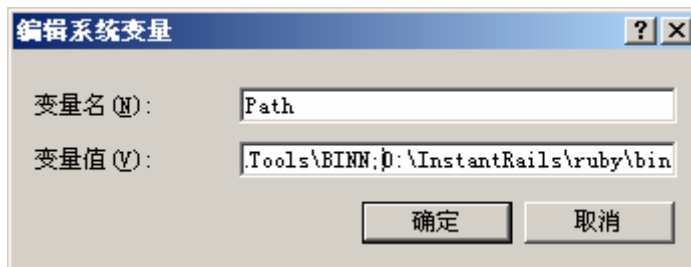
- REST Intro
- REST from the Distributed System Perspective
- Architecture of REST
- REST on Rails**
- Consuming RESTful Web Service
- Tech Community
- Next Time

□ Task 1: 生成REST框架

```
cd RESTDemo
```

```
ruby script/generate scaffold_resource product name:string price:double
```

- 注意：要将ruby/bin目录放入path中



```
class PeopleController < ActionController::Base
  def index
    @people = Person.find(:all)

    respond_to do |format|
      format.html # renders index.rhtml
      format.js   # renders index.rjs
      format.xml  { render :xml => @people.to_xml }
      format.icl { render_calendar(@people) }
      format.atom do
        render :action => "atom", :content_type => Mime::ATOM
      end
    end
  end
end
```

```

class PeopleController < ActionController::Base
  def index
    @people = Person.find(:all)

    respond_to do |format|
      format.html # renders index.rhtml
      format.js   # renders index.rjs
      format.xml  { render :xml => @people.to_xml }
      format.icl  { render_calendar(@people) }
      format.atom do
        render :action => "atom", :content_type => Mime::ATOM
      end
    end
  end
end
end
end

```

GET /people
=> returns HTML

GET /people.xml
=> returns XML

Accept: text/javascript
GET /people
=> returns RJS

Accept: text/html
GET /people.xml
=> returns XML

```

class PeopleController < ActionController::Base
  def index
    @people = Person.find(:all)

    respond_to do |format|
      format.html # renders index.rhtml
      format.js   # renders index.rjs
      format.xml  { render :xml => @people.to_xml }
      format.icl  { render_calendar(@people) }
      format.atom do
        render :action => "atom", :content_type => Mime::ATOM
      end
    end
  end
end
end
end

```

```

GET /people
=> returns HTML

```

```

GET /people.xml
=> returns XML

```

```

Accept: text/javascript
GET /people
=> returns RJS

```

```

Accept: text/html
GET /people.xml
=> returns XML

```

```

class PeopleController < ActionController::Base
  def index
    @people = Person.find(:all)

    respond_to do |format|
      format.html # renders index.rhtml
      format.js   # renders index.rjs
      format.xml  { render :xml => @people.to_xml }
      format.icl  { render_calendar(@people) }
      format.atom do
        render :action => "atom", :content_type => Mime::ATOM
      end
    end
  end
end
end
end

```

GET /people
=> returns HTML

Accept: text/javascript
GET /people
=> returns RJS

GET /people.xml
=> returns XML

Accept: text/html
GET /people.xml
=> returns XML

```

class PeopleController < ActionController::Base
  def index
    @people = Person.find(:all)

    respond_to do |format|
      format.html # renders index.rhtml
      format.js   # renders index.rjs
      format.xml  { render :xml => @people.to_xml }
      format.icl  { render_calendar(@people) }
      format.atom do
        render :action => "atom", :content_type => Mime::ATOM
      end
    end
  end
end
end
end

```

GET /people
=> returns HTML

Accept: text/javascript
GET /people
=> returns RJS

GET /people.xml
=> returns XML

Accept: text/html
GET /people.xml
=> returns XML

- REST Intro
- REST from the Distributed System Perspective
- Architecture of REST
- REST on Rails
- Consuming RESTful Web Service
- Tech Community
- Next Time

□ Task 2 使用Ruby调用RESTful Web Service

```
require 'rexml/document'
require 'open-uri'

f = open("http://localhost:3006/products.xml")
products = REXML::Document.new(f)

products.root.each_element do |product|      # each <product> in <products>
  product.each_element do |node|           # <name>, <price>, etc. in <product>
    # the contents of <name>, <price>, etc.
    puts "#{node.name}: #{node.text}"
  end
end
end
```

- 附加题： 使用Java调用RESTful Web服务
- 修改第4次实验的代码，使用Dom4j解析今天的RESTful Web Service，并在终端里打印解析后的内容。

- REST Intro
- REST from the Distributed System Perspective
- Architecture of REST
- REST on Rails
- Consuming RESTful Web Service
- **Tech Community**
- Next Time

Tech Community



□ InfoQ.com



□ TheServerSide.com



□ JavaEye.com



- REST Intro
- REST from the Distributed System Perspective
- Architecture of REST
- REST on Rails
- Consuming RESTful Web Service
- Tech Community
- Next Time