RESTful 架构基础

ImportNew 方志朋 10月21日

> 来自：唐尤华
> 译自：https://dzone.com/refcardz/rest-foundations-restful

REST（Representational State Transfer）架构风格是一种世界观，把信息提升为架构中的一等公民。通过 REST 可以实现系统的高性能、可伸缩、通用性、简单性、可修改性和可扩展等特性。这篇文章解释了主要的 HTTP 操作，对 HTTP 响应码进行描述，并列举相关开发库和框架。此外，本文还提供了额外的资源，对每个主题进行了更深入的探讨。

1. 简介

REST 架构风格不是一种可以购买的技术，也不是一个可以添加到软件开发项目中的开发库。首先也是最重要的，REST 是一种世界观，把将信息提升为构建架构中的一等公民。

Roy Fielding 的博士论文“架构风格和基于网络的软件架构设计”介绍和整理了“RESTful”系统的思想和相关术语。这是一篇学术论文，虽然使用正式语言，但是仍然易于理解并且提供了实践基础。

总结一下，RESTful 通过体系结构的特定选择能从部署的系统中获得理想特性。尽管这种风格定义的约束细节并没有为所有场合设计，但是的确可以广泛适用。

由于 Web 对消费者偏好有多重影响，REST 风格的倡导者鼓励企业组织在其边界内使用相同原则，就像他们在面向外部客户的网页上做的那样。本文将讨论现代 REST Web 实现中的基本约束和属性。

1.1 基础概念

REST 表示什么含义？以无状态方式传输、访问和操作文本数据。当正确部署后，REST 为互联网上不同应用程序之间提供了一致的互操作性。无状态（stateless）这个术语至关重要，它使得应用程序可以用不可知的方式进行通信。RESTful API 通过统一资源定位符地址（URL）公开服务。URL 名称将资源的区分为接受内容或返回内容。RFC 1738 中定义了 URL scheme，可以在这里找到：https://tools.ietf.org/rfc/rfc1738.txt

RESTful URL 类似于下面这个 library API：

```
http://fakelibrary.org/library
```

实际公开的不一定是某种任意的服务，而是代表对消费者有价值的信息资源。URL 作为资源句柄，可以请求、更新或删除内容。

开始把服务发布到某个地方，然后开始与 REST 服务进行交互。返回的内容可能是 XML、JSON 格式，或者更确切地说是像 Atom 或自定义 MIME 类型等超媒体格式。虽然一般建议尽可能重用现有的格式，但是对正确设计的媒体类型正在变得越来越宽容。

需要请求资源的时候，客户机会发一个超文本传输协议（HTTP）GET 请求，例如在浏览器中键入一个 URL 然后点击回车，选择书签，或者点击锚引用链接。
通过编程方式与 RESTful API 交互，有数十个客户端 API 或工具可供选择。使用 curl 命令行工具，可以输入以下命令：

```
$ curl http://fakelibrary.org/library
```

上面的命令使用默认格式，但你可能不需要这种格式的信息。幸运的是 HTTP 有一种机制，可以指定返回信息的格式。在请求中指定 "Accept" 头，如果服务器支持这种格式，会以指定的格式返回。这个过程称为内容协商，这是 HTTP 中未被充分利用的功能之一，可以使用一个类似于上面例子中的 curl 命令来指定：

```
$ curl -H "Accept:application/json" http://fakelibrary.org/library
```

由于资源名称与内容格式是独立的，从而让请求不同格式信息成为可能。虽然 REST 中的 “R” 的含义是“表现”而非“资源”，但是应该在构建系统时允许客户端指定请求的内容格式，请牢记这一点。在我们的例子中 library API 可能包含以下 URL：

- http://fakelibrary.org/library：图书馆基本信息，搜索图书、DVD等相关资源基本功能的链接。
- http://fakelibrary.org/book：存放书籍的“信息空间”。从概念上说，这里可能会存放所有的书籍。显然，如果这个问题得到解决，我们不会希望返回所有图书，而是希望通过类别、搜索关键词等来检索图书。
- http://fakelibrary.org/book/category/1234：在书籍的信息空间里，我们可以指定类别浏览，例如成人小说、儿童书籍、园艺书籍等。使用杜威十进制图书分类法是可行的，但我们也可以想象自定义分组。问题的关键在于，这种“信息空间”可能是无限的，而且可能收到人们实际关心的信息类型影响。
- http://fakelibrary.org/book/isbn/978-0596801687：提到某本具体的书，应该包括书名、作者、出版商、系统中的拷贝数、可用拷贝数等信息。

> 译注：杜威十进制图书分类法由美国图书馆专家麦尔威·杜威发明，于1876年首次发表，历经22次的大改版。该分类法以三位数字代表分类码，共可分为10个大分类、100个中分类及1000个小分类。

就图书馆用户而言，上面提到的这些 URL 可能就是只读的，但是图书馆员使用应用程序时实际上可以操作这些资源。

例如添加一本新书，可以向 main/book 地址 POST 一个 XML。使用 curl 提交，看起来可能像这样：

```
$  curl -u username:password -d @book.xml -H "Content-type: text/xml" http://fakel
```

此时，服务器可能会对提交的内容进行校验，创建与图书相关的记录，并返回响应代码201——表示已创建新资源。新资源的 URL 可以在响应的 Location 头中找到。

RESTful 请求一个重要特性：每次请求都包含了充足的状态信息来响应请求。这为服务器的可见性和无状态创造了条件，并为扩展系统和识别发送的请求内容提供了理想特性。对于缓存结果也非常有帮助。服务器地址和请求状态组合成可计算的 hash 键值，并形成一个结果集：

```
http://fakelibrary.org + /book/isbn/978-0596801687
```

接下来我们会先介绍 GET 请求。客户端在需要时发出 GET 请求获取指定资源。客户端可以在本地缓存请求结果，服务器可以在远程缓存结果，系统的中间层可以在请求链路中间缓存结果。这是一个与具体应用程序无关的特性，可以加入系统设计中。

正因为可以操作资源，也就意味着并不是每个人都可以这样做。我们完全可以建立一个防护模型，要求用户在操作前验证身份，证明他们具有该操作的授权。在本文的最后，将提供一些提升 RESTful 服务安全性的内容。

2. REST 和 SOAP 比怎么样?

> SOAP：简单对象访问协议（Simple Object Access Protocol）。是交换数据的一种协议规范，是一种轻量的、简单的、基于XML的协议。一条 SOAP 消息就是一个普通的 XML 文档，包含必需的 Envelope 元素、可选的 Header 元素、必需的 Body 元素和可选的 Fault 元素。

把 REST 与 SOAP 划等号是错误的。在这两者之间进行比较，带来的困扰远多于好处。简单来说，它们不是一回事。尽管可以用这两种方法解决许多架构问题，但是它们不能相互替换。

这种混淆很大程度上源于对 “REST 是通过 URL 调用 Web 服务”这句话的误解。这种观点与 RESTful 架构的功能相距甚远。如果不全面深入理解 RESTful 的架构实现，就很容易误解 REST 实践的本意。

利用 REST 的最佳方式，是将生产和消费过程中的信息与技术分离实现解耦，进而更好地管理系统，让架构具备以下特性：

- 高性能
- 可扩展

- 通用
- 简洁
- 可修改
- 可扩展

这并不是说，基于 SOAP 构建的系统不能具备上述特性。而是当技术、组织或过程的复杂性造成不能在单个事务中完成请求的生命周期时，这种情况 SOAP 能够发挥最佳效果。

3. Richardson 成熟度模型

Leonard Richardson 引入了一种成熟度模型，部分阐述了 SOAP 与 REST 之间的区别，并提供一种对不同类型的系统进行分类的框架。许多人不恰当地称之为 “REST”。可以将这种分类看作系统中不同 Web 技术组件紧密程度的度量标准：包括信息资源、HTTP 作为应用层协议和作超媒体作为控制媒介。

称其为“成熟度模型”似乎意味着应该只构建“成熟度”最高的系统。这种看法是不合适的。第2级是有价值的，从2级向3级转变通常只是采用了一种新的 MIME 类型。然而，从0级到3级的转变要困难得多，因此增量式升级转变通常也会增值。

首先，确定希望公开哪些信息资源。采用 HTTP 作为处理这些信息资源的应用协议，包括内容协商。接下来，当一切就绪时，使用基于超媒体的 MIME 类型，这样就可以充分享受 REST 的好处了。

4. 动词

动词是用来与服务器资源交互的方法或操作。RESTful 系统中有限的动词让刚接触该的使用者感到困惑和沮丧。看似武断和不必要的约束，目的是鼓励以应用程序无关的形式提供可预测的行为。通过明确、清晰地定义这些动词的行为，客户端可以在网络中断或故障时自主处理。

精心设计的 RESTful 系统主要使用4个 HTTP 动词。

4.1 GET

GET 请求是最常用的 Web 动词。GET 请求将命名资源从服务器传输到客户端。尽管客户端不需要知道请求的资源内容，但是请求返回的结果是带元数据标记的字节流，这表明客户端应该知道如何解释资源。在 Web 中通常用 “text/html” 或 “application/xhtml+xml” 表示。正如之前提到的那样，只要服务器支持，客户端可以通过内容协商提前指定请求的返回格式。

GET 请求关键点之一，不要修改服务器端的任何内容。这是一个基本的安全要求，也是不熟悉 REST 的开发者犯的最大错误之一。你可能会遇到这样的 URL：

```
http://example.com/res/action=update?data=1234
```

不要这样做！ 由于 GET 请求安全性允许缓存请求，这会让正在构建的 RESTful 系统陷入混乱。GET 请求也意味着幂等性，即多次请求不会对系统产生任何影响。这是基于分布式基础设施的一个重要特性。如果进行 GET 请求时被打断，由于幂等性，客户端可以再次发起请求。这点非常重要。在设计良好的基础结构中，客户端可以从任意应用程序发起请求。虽然一定会有与应用程序相关的特定行为，但是加入与应用程序无关的行为越多，系统就会越有弹性，也更容易维护。

4.2 POST

在辨别 POST 和 PUT 动词意图的时候，情况开始变得不那么清晰。根据定义，二者似乎都可以被客户端用来创建或更新服务器资源，然而它们的用途各有不同。

当无法预测请求创建的资源的标识时，客户端会使用 POST 请求。在新增雇员、下订单或提交表单的时候，我们无法预测服务器将如何命名正在创建的资源。这就是为什么将资源提交给类似 Servlet 这样的程序处理。接下来，服务器会接受请求、校验请求、验证用户凭据等。成功处理后，服务器将返回 201 HTTP 响应代码，其中包含一个 “Location” 头，代表新创建的资源的位置。

注意： 有些人将 POST 视为创建资源的 GET 会话。他们会对创建的资源通过 body 返回200，而不是返回201。这似乎是避免二次请求的一种快捷方式，但是这种做法混合了 POST 和 GET，让缓存资源的潜在影响变得微妙。尽量避免因为走捷径而牺牲大局。短期看这似乎是值得的，但随着时间的推移，这些捷径叠加起来可能会带来不利的影响。

POST 动词的另一个主要用途是“追加（Append）”资源信息，即增量编辑或部分更新，而不是提交完整的资源。这里应使用 PUT 操作。对已知资源使用 POST 更新，可用于向订单添加新送货地址或更新购物车中某个商品的数量。

由于是更新资源的部分信息，**POST 既不安全也不幂等。**

POST 的最后一种常见用法是提交查询。将查询的内容或表单内容进行 URL 编码后提交给服务执行查询。通常可以直接返回 POST 结果，因为没有与查询相关的标识。

注意： 建议将这样的查询转换为信息资源本身。如果采用 POST 查询，可以考虑采用 GET 请求，后者支持缓存。你可以与其他人分享这个链接。

4.3 PUT

由于 HTML 表单目前还不支持 PUT，许多开发人员基本上会忽略 PUT 动词。然而，PUT 有一个重要作用并且是 RESTful 系统完整愿景的一部分。

客户端可以向指定 URL 发 PUT 请求，服务器用请求中的数据执行覆盖操作。PUT 请求在某种程度上是等幂的，而 POST 更新不是。

如果客户端在 PUT 覆盖请求时被打断，由于重新发送覆盖操不会造成任何后果，因此可以再次发送。客户端具备管理状态能力，所以直接重发覆盖命令即可。

注意： 这种协议层处理并不意味着要取消更高级别（如应用层）的事务，但是同样地，它也是一种体系结构上理想的属性，可以在应用层以下使用。

如果客户端能够提前了解资源的标识，那么 PUT 也可用于创建资源。正如我们在 POST 部分中讨论的那样，通常不会出现这种情况。但是如果客户端能够控制服务器端信息空间，那么这种操作也是合理的。

4.4 DELETE

在公共网络上 DELETE 动词没有被广泛使用（谢天谢地!）。然而，对于控制信息空间非常有用，它是资源生命周期中非常有用的一部分。

DELETE 请求意在实现等幂。可能由于网络故障 DELETE 请求被打断，这时我们希望客户端继续尝试。第一次请求无论成功与否，资源都应该返回204（无指定内容）。对之前已删除的资源或不存在的资源可能需要一些额外处理，两种情况都应该返回404。一些安全策略要求为不存在的和已删除的资源返回404，这样 DELETE 请求就不会泄漏有关资源是否存在的信息。

还有另外三个没有广泛使用但是有价值的动词。

4.5 HEAD

HEAD 动词用来请求资源，但不实际检索。客户端可以通过 HEAD 检查资源是否存在，并检查资源相关的元数据。

4.6 OPTIONS

OPTIONS 动词也可以用来查询服务器相关资源的情况，方法是询问哪些其它动词可用于该资源。

4.7 PATCH

最新的动词 PATCH 直到2010年才正式采纳为 HTTP 的一部分。旨在提供一种标准化方式来表示部分更新。PATCH 请求通过标准格式让交互的意图更明确。这是推荐使用 PATCH 而非 POST 的原因，尽管 POST 可以用于任何事情。IETF 发布了 RFC 文档，定义用于 PATCH 操作的 XML 和 JSON。

如果客户端 PATCH 请求的 header 中带 If-Match，则此部分为幂等更新。可以重试中断的请求，因为如果第一次请求成功，那么 If-Match header 会不同于新状态。如果相同，则未处理原始请求可应用 PATCH。

5. 响应码

HTTP 响应码为我们在客户端和服务器之间的对话提供了丰富的请求状态信息。大多数人只熟悉一般意义上的200、403、404或者500，但是还有更多有用的代码可供使用。这里表格并不全面，但是它们涵盖了许多在 RESTful 环境中应该考虑使用的最重要代码。数字可按照以下类别分组：

- 1XX：信息类
- 2XX：操作成功
- 3XX：重定向
- 4XX：客户端错误

- 5XX：服务器错误

第一组响应码表明客户端的请求格式正确且处理成功。具体操作如下表所示：

响应代码	描述
200	OK。请求已成功执行，回应内容取决于所调用的动词。
201	已创建。请求已成功执行，在执行过程中创建了一个新资源。响应 body 为空或包含所创建资源的 URI。响应中的 Location 头也应该指向 URI
202	已接受。请求有效并已接受，但尚未得到处理。可通过请求轮询，响应结果中 URI 提供状态更新。这种方式支持异步 REST 请求
204	没有请求的内容。请求已成功处理，但服务器没有任何响应结果，客户端不应更新显示

表1 成功的客户端请求

响应代码	描述
301	永久移除。请求的资源不再位于指定的 URL，新的 Location 应该在响应头中返回。只有 GET 或 HEAD 请求应当重定向到新位置。如果可能，客户端应该更新自己保存的书签
302	已找到。请求的资源已在其它临时位置找到，并应该在响应头中返回该位置。只有 GET 或 HEAD 请求应该重定向到新位置。客户端无需更新书签，因为资源会返回对应的 URL。
303	参见其他信息。该响应码被 W3C 技术架构小组（TAG）重新解释成一种对非网络可寻址资源的有效请求方式。这是语义化 Web 中的一个重要概念，可以向个人、概念、组织等提供 URI。能够在 Web 中找到和不能在 Web 中找到的资源是有区别的。如果客户端收到的响应码是303而不是200，就能分辨这种差别。重定向的位置在响应的 Location 头中。头信息可能包含相关资源的文档引用，也可能包含相关资源的一些元数据

表2 — 客户端重定向请求

表3中的响应代码表示客户端请求无效，如果条件不发生变化，重新请求仍无法处理。这些故障可能有请求格式错误、未授权的请求、请求的资源不存在等。

响应代码	描述
405	方法不允许
406	请求不接受
410	资源不存在
411	需要指定长度
412	前提条件失败
413	Entity 太大
414	URI 超长
415	不支持的媒体类型
417	预期失败

表3 客户端请求错误

最后，表4中的响应代码表示服务器暂时无法处理客户端请求（可能仍然无效）。客户端应当在将来的某个时候重新请求。

响应代码	描述
500	内部服务器错误
501	未实现
503	服务不可用

表4 服务器处理请求错误

服务根据其自身功能要求具有不同程度的可扩展性。

注意： 试试响应代码418，它会返回简洁有力的回复："我是一个茶壶。"

5.1 REST 资源

5.1.1 论文

Fielding 博士的论文《架构的风格与基于网络的软件架构设计》是对 RESTful 思想的主要介绍：http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

5.1.2 RFC 规范

REST 常见用法的技术规范由**国际互联网工程任务组（IETF）定义，按照请求评议（RFC）**流程完善。规范由数字定义，并随着时间推移不时更新版本，以替换已经过时的文件。目前，这里有最新的相关 RFC 文件。

5.1.2.1 URI

RFC 3986定义了 URI 命名方案的通用语法。URI 是一种命名方案，包含了对其他如网址、支持名字子空间等编码方案。网址：http://www.ietf.org/rfc/rfc3986.txt>

5.1.2.2 URL

Url 是 URI 的一种形式，其中嵌入了充足的信息（通常是访问方案和地址），用于解析和定位资源统一资源定位符。网址：http://www.ietf.org/rfc/rfc1738.txt

5.1.2.3 IRI

国际化资源标识符（IRI）在概念上是一个用 Unicode 编码的 URI，用于在 Web 上使用的标识符中支持世界上各种语言的字符。IETF 选择创建一个新的标准，而不是改变 URI 方案本身，以避免破坏现有的系统并明确区分这两种方法。那些支持 IRI 的人故意这样做。还定义了在 IRI 和 URI 之间进行转换的映射方案。网址<：http://www.ietf.org/rfc/rfc3987.txt

5.1.2.4 HTTP

HTTP 1.1版本定义了一个应用程序协议，用于操作通常以超媒体格式表示的信息资源。虽然它是一个应用级协议，但通常不与应用程序绑定，由此产生了重要的体系结构优势。大多数人认为 HTTP 和超文本标记语言文（HTML）就是“Web”，但是 HTTP 在非面向文档的系统开发中也很有用。网址：http://www.ietf.org/rfc/rfc2616.txt

5.1.2.5 PATCH 格式

JavaScript 对象表示法（JSON）Patch 网址：https://www.ietf.org/rfc/rfc6902.txt
XML Patch 网址：https://www.ietf.org/rfc/rfc7351.txt

5.2 描述语言

人们对使用各种语言来描述 API 非常感兴趣，通过描述语言可以更容易地编写客户端和服务器文档，甚至生成骨架代码。一些比较流行、有趣的描述语言包括：

5.2.1 RAML

RAML 是一种 YAML/JSON 语言，可以定义2级成熟度的 API。它支持可重用模式和特性，通过模式和特性实现功能 API 设计的标准化。网址：http://raml.org

5.2.2 Swagger

Swagger 是另一种 YAML/JSON 语言，支持定义2级成熟度的 API。它包含代码生成器、编辑器、API 文档可视化功能，能够与其他服务集成的。网址：http://swagger.io

5.2.3 Apiary.io

Apiary.io 是一个协作式的托管站点。它支持 Markdown 格式的 API 文档，可以围绕设计过程进行社交，并且支持模拟数据的托管实现，以便于在 API 实现之前对其进行测试。网址：http://apiary.io

5.2.4 Hydra-Cg

Hydra-Cg 是一种超媒体描述语言，通过像 JSON-LD 这样的标准方便地实现数据关联和并其它数据源的交互。网址：http://www.hydra-cg.com

5.3 实现

有一些用于构建、生成和使用 RESTful 系统的库和框架。虽然任何 Web 服务器都可以配置成提供 REST API，但有了这些框架、库和环境可以让过程变得更容易。

以下概述了一些主流的环境：

5.3.1 JAX-RS

JAX-RS 规范为 JEE 环境增加了对 REST 的支持。网址：https://jax-rs-spec.java.net

5.3.2 Restlet

Restlet API 是构建用于生产和消费 RESTful 系统的 Java API 先行者之一。它专注于为客户端和服务器生成一些非常干净、强大的 API。

Restlet Studio 是一个免费工具，能够在 RAML 和基于 swagger 的 API 描述之间进行转换，支持 Restlet、 Node 和 JAX-RS 服务器和客户端的骨架和 Stub 代码。网址：http://restlet.org

5.3.3 NetKernel

Netkernel 是一个比较有趣的 RESTful 系统。它基于微内核，是支持各种架构风格环境的代表。Netkernel 受益于在软件体系结构中采用 Web 的经济属性。你可以把它想象成“在内部引入 REST”。虽然任何基于 REST 的系统在外面看起来都一样，但在运行环境内部 NetKernel 看起来也一样。网址：http://netkernel.org

5.3.4 Play

两个主要的 Scala REST 框架之一。网址：https://www.playframework.com

5.3.5 Spray

两个主要的 Scala REST 框架之一。它设计成配合 Akka actor 模型一起工作。网址：http://spray.io

5.3.6 Express

两个主要的 Node.js REST 框架之一。网址：http://expressjs.com

5.3.7 hapi

两个主要的 Node.js REST 框架之一。网址：http://hapijs.com

5.3.8 Sinatra

Sinatra 是一个领域特定语言（DSL），用来在 Ruby 中创建 RESTful 应用程序。网址：http://www.sinatrarb.com

5.4 客户端

通过浏览器调用 REST API 是可行的，但是还有其它客户端可用于测试和构建面向资源的系统。

5.4.1 curl

curl 是流行的库和命令行工具之一，支持在各种资源上调用各种协议。网址：https://curl.haxx.se

5.4.2 httpie

httpie 是一个非常灵活和易用的客户端，支持通过 HTTP 与资源进行交互。网址：https://httpie.org

5.4.3 Postman

健全的 API 测试需要能够捕获和重播请求，支持各种身份验证和授权方案等功能。以前的命令行工具允许这样做，但 Postman 是一个较新的桌面应用程序，让这些工作对于开发团队来说变得更容易。网址：https://www.getpostman.com

6. 书籍

- “RESTful Web APIs”：Leonard Richardson、Mike Amundsen 和 Sam Ruby，2013，O’Reilly 出版社
- “RESTful Web Services Cookbook”：Subbu Allamaraju，2010，O’Reilly 出版社
- “REST in Practice”：Jim Webber、Savas Parastatidis 和 Ian Robinson，2010，O’Reilly 出版社。中文版《REST实战(中文版)》
- “Restlet in Action” by Jerome Louvel and Thierry Boileau，2011，Manning 出版社
- “Resource-Oriented Architecture Patterns for Webs of Data (Synthesis Lectures on the Semantic Web: Theory and Technology)”：Brian Sletten，2013，Morgan & Claypool

- 最近面试Java后端开发的感受
- 8种常见SQL错误用法
- 分布式 id 生成器
- Java 程序员必须清楚的 7 个性能指标

最近面试BAT，整理一份面试资料《**Java面试BAT通关手册**》，覆盖了Java核心技术、JVM、Java并发、SSM、微服务、数据库、数据结构等等。

获取方式：点“**在看**”，关注公众号并回复 **666** 领取，更多内容陆续奉上。

明天见(。·ω·。)ﾉ♡