

Mashups——一种新型 Web 应用程序

凌妍妍 (Web 组)

1. 引言

一种新型的基于 Web 的数据集成应用程序正在 Internet 上逐渐兴起。通常用术语 mashup 表示。根据 Wikipedia 的解释, Mashup 是将多个不同的支持 web api 的应用进行堆叠而形成的新型 web 服务。这种新型的基于 Web 的数据集成应用程序正在 Internet 上逐渐兴起。它利用了从外部数据源检索到的内容来创建全新的创新服务,将来自不止一个数据源的内容进行组合,创造出更加增值的服务。Mashup 所能利用的外部数据源格式多种多样,表现出惊人的兼容性,它涵盖 public APIs, XML/RSS/Atom feeds, web services, HTML 等。人们普遍认为 Mashup 具有 Web 2.0 的特点。Web 2.0 的主要思路是在互联网上建立起大众的贡献的共享的信息平台,协作和共享是这种思想的精髓。Mashup 技术也是建立在各种 Web 应用程序贡献出自己的服务和内容,同时共享其他人和其他组织提供的信息和服务的基础上的,自此基础上进行组合、增值从而构造出更多更具吸引力的新的 Web 应用程序。随着越来越多的 Web 站点公开了自己的 api,许多人已经和正在用 eBay, Amazon, Google and Yahoos APIs 构建新的 Mashups,使得这种新型的 Web 应用模式成为了现实。这篇文章对 Mashup 的分类和架构进行了深入的研究和探索,另外您还将看到对 Mashup 在企业应用中引出的研究问题的一些分析。

2. Mashup 分类

本节将简要介绍[2]中对出名的 Mashup 类型进行调查的一些成果。现今涌现的 Mashup 应用大致由以下几类构成:

视频图像 Mashup——Mashup 设计者利用与图像相关的元数据(例如谁拍的照片,照片的内容是什么,何时何地拍摄的等)对视频和图像资源进行关联。CelebrityV 就是这样的应用——它将来自 Flickr 的明星照片和来自 youtube 对应的明星视频剪辑进行了匹配和组合。

搜索购物 Mashup——搜索和购物 mashup 在 mashup 这个术语出现之前就已经存在很长时间了。在 Web API 出现之前,有相当多的购物工具,例如 BizRate、PriceGrabber、MySimon 和 Froogle,都使用了 B2B 技术或屏幕抓取(screen scraping)的方式来累计相关的价格数据并进行比较。之后为了促进 mashup 和其他有趣的 Web 应用程序的发展,诸如 eBay 和 Amazon 之类的消费网站已经为通过编程访问自己的内容而发布了自己的 API。

新闻 Mashup——新闻源(纽约时报、BBC 或路透社)已从 2002 年起使用 RSS 和 Atom 之类的联合技术来发布各个主题的新闻提要。以联合技术为基础的 mashup 可以聚集一名用户的提要,创建个性化的报纸,从而满足读者独特的兴趣。Diggdot.us 正是这样一个应用。

地图 Mashup——地图 Mashup 蓬勃发展的一种主要动力就是 Google 公开了自己的 Google Maps API。现阶段,几乎所有包含位置数据的数据集均可利用地图通过令人惊奇的图形化方式呈现出来。Microsoft (Virtual Earth)、Yahoo (Yahoo Maps) 和 AOL (MapQuest) 也不甘示弱,很快相继公开了自己的 API。

3. Mashup 架构

此架构[3]非常简单。来自客户浏览器的请求传向 Mashup 站点所在的 Apache Web 服

务器。请求的页面包括 HTML 和 JavaScript。JavaScript 调用一个或多个 API 内容提供者提供的服务后,按照该 Mashup 的逻辑进行内容组合。举一个 Mashup 的例子,用户向 Mashup 站点提交房屋的所在地点和房价查询是否高于当地平均水平,一个请求就传向一个与后台房屋信息数据库连接的 Web 服务器,同时调用 Google Maps API 提供的服务,执行 Mashup 逻辑并将组合的内容在客户机端浏览器中显示。一般来说, Mashup 服务主要涉及 3 方面的内容: Mashup 站点, API 内容提供者以及客户机的 Web 浏览器。

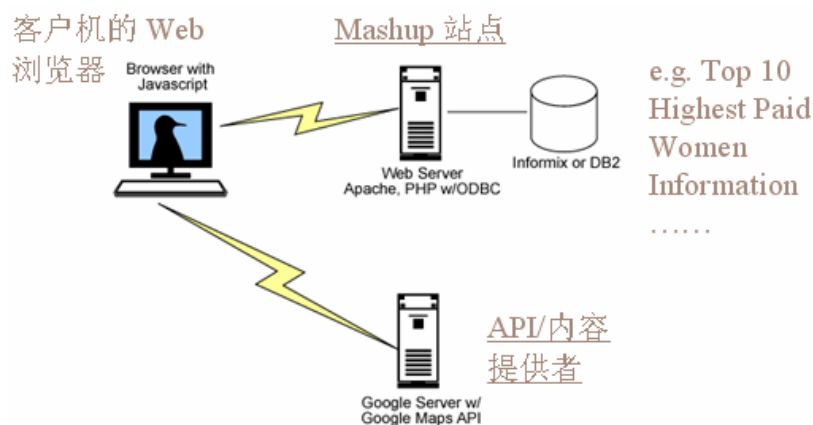


图 1: Mashup 架构

3.1 Mashup 站点

一个 Mashup 应用可能需要将来自多个数据源的信息进行合并组合,构造新的服务,因而这里所说的 Mashup 站点也就是 Mashup 逻辑所在的地方。尽管 Mashup 这类新型的 Web 应用程序可以采用以往的 Web 服务器技术 (Java servlets、CGI、PHP 或 ASP) 构造传统 Web 应用程序, 我们却发现越来越多的 API 开始设计成通过浏览器端的 JavaScript 进行访问。于是对于 Mashup 的执行来说, 合并内容也可以直接在客户机的浏览器中通过客户机端脚本 (即 JavaScript) 或 applet 生成。我们将 Mashup 使用的这种方法称为胖 Internet 应用程序 (简称 RIA)。

在客户机端进行 Mashup 应用中的内容合并, 其优点可以概括如下: 首先, 从 Mashup 服务器的角度来说, 对服务器的所产生的负载较轻 (数据可以直接从内容提供者那里传送过来); 其次, 从用户的角度来说, 具有更好无缝用户体验 (页面可以请求对内容的一部分进行更新, 而不用刷新整个页面), 当然这样的功能得益于 Ajax 这样的 Web 应用模型的诞生。

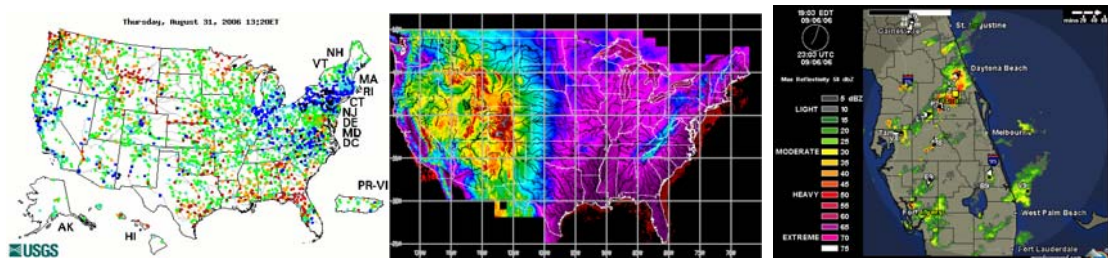
3.2 API 内容提供者

他们提供的内容为 Mashup 应用程序所用。为了方便外界获取和使用, 他们将自己的内容通过 Web 协议对外提供 (例如 REST、Web 服务和 RSS/Atom)。按照其通常的功能和使用, Web 协议可以分为两组。第一组处理消息传递、接口描述、寻址和交付的问题。最有名的是消息传递协议, 称为简单对象访问协议 (Simple Object Access Protocol, SOAP)。此协议对消息进行了编码, 这样就可以通过传输协议 (如 HTTP、IIOP、SMTP 或其他协议) 在网络上传递它们。下一组协议和规范定义了服务如何公开它们自己以及如何在网上相互发现。对于要相互查找的服务, 统一描述、发现和集成 (Universal Description, Discovery and Integration, UDDI) 为查找和访问服务定义了注册中心和相关的协议。当然, Web 上还存在着很多有趣的潜在数据源可能并没有方便地对外提供 API, Mashup 应用如果想用到这些信息, 可以通过前面我们提到的屏幕抓取的技术实现, 通过对特定页面进行分析, 提取 Mashup 感兴趣的信息。

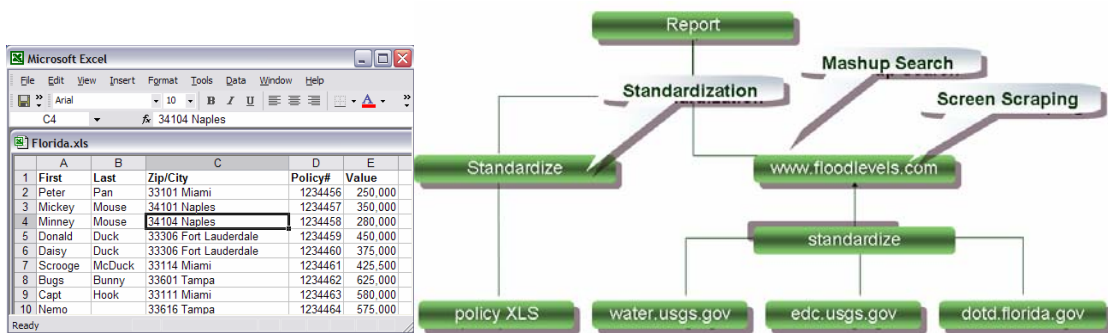
4. 企业级 Mashup 及引出的研究问题

Mashup 技术并非只会提供消费者网站使用的、加了注释的地图，这项技术具有真正的企业应用前景。Mashup 这样轻量级的集成在企业中有着很多的先例：从历史悠久的股票报价系统，到将 UPS 或 FedEx 等快递公司的跟踪数据与订单记录组合起来，提供订单状态单一视图的电子商务网站等等。IBM 等门户服务器厂商提供了诸如 StrikeIron 之类的图形化操作工具，帮助用户集成来自不同地方的数据源，实现简单、个性化的 Web 应用。

一方面，企业必须将许多原本并不能很好彼此共存的管理系统和应用程序拼凑到一起。DBMS、内容管理系统、数据挖掘包和工作流系统都可以购买，但该公司必须自行开发集成软件以集成它们。每当增加了新的数据源或信息必须流转到新的目标时，就必须扩展客户自制的解决方案。因此，企业需要一个提供所有这些服务的统一视图的健壮平台，这样的平台应该突破存在于 DBMS、内容管理系统、中间层高速缓存和数据仓库之间的界限。另一方面，[4]中指出即时应用的出现使得利用企业信息架构之外的信息成为新的需求（电子邮件，报告和文档，网页，电子表格，决策支持数据等等）。这样的即时应用对效率和易用性都提出了新的要求，因此在企业级 Mashup 应用中出现了用“assembly”来代替“programming”的思想。



(HUC = Hydrological Unit Code) (Geocode = Latitude/Longitude) (Geocode = Latitude/Longitude)



(Zipcode)

图 2: 构造 Mashup 的过程

首先来看一个企业即时应用的实例（例 1）——佛罗里达州的一名保险经纪看见了一则美国风暴灾害的新闻报道，公司要求他递交一份最新灾害损失分析报告，对这场风暴给公司带来的损失进行评估。如图 2 所示，该名保险经纪为了完成这份受灾情况分析报告，他首先需要将电子表格里客户的 ZipCode 通过网站 Water.usgs.gov 转化为水文学上的一种编码，然后利用网站 edc.usgs.gov 将水文学编码转化为经纬度的表示，最终从网站 dotd.florida.gov 上通过经纬度定位当地的受灾情况。当然，如果 Web 上已经存在了这样一个 Mashup 应用 www.floodlevels.com 合并了来自上述三个网站的内容和服务，即直接提供了从 ZipCode 查询受灾情况的功能。那么如此现存可利用的资源更应该得到有效发掘和运用。

下面我们将从构造企业级 Mashup 应用的四个方面分别讨论存在的研究问题。

● 资源发掘

构造企业级 Mashup 应用并不一定是从零做起，Web 上存在着许多对处理企业即时应用有效的资源和服务。比如，例 1 中提到的网站 www.floodlevels.com 就是一个现存的可满足从 ZipCode 查询受灾情况的 Mashup 站点，它隐藏在 Web 信息海洋中。可想而知，利用现存可利用的 Mashup 作为构建企业解决方案的基础，在效率和有效性上都会带来质的飞跃，也大大省去了 Mashup 应用重复开发带来的资源浪费。

于是一种新形式的搜索引擎正在酝酿。这种搜索引擎试图通过使用户键入简单的查询，来发掘 Web 中所有与该查询相关的 Mashup 资源。如使用者期望通过查询“Flood Levels”来找到这样的一个 Mashup 资源 www.floodlevels.com 为我所有。Mashup 资源发掘的问题和 Deep Web 研究领域特定类别数据源的发现问题有异曲同工之处。相关资源的发掘需要解决两方面的问题，一是对资源特性的描述，而是资源与用户需求的匹配程度。Deep Web 数据源的特点相对来说比较明确，我们可以从接口的复杂性，结果的结构化等来对 Deep Web 数据源进行定义并分析该数据源的类别和接口形式。但是对于 Mashup 数据源来说，最大的困难是如何让搜索引擎理解 Mashup 的逻辑，从而推断与用户需求的匹配程度。

● 需求表达

构造企业级 Mashup 应用的第二阶段是用户的需求表达，这主要包括两方面的含义：首先，如果在资源发掘阶段能够找到现存的可直接利用的 Mashup 很好地解决用户的需求，那么用户可以在此基础上进行上层应用的构建。比如一旦 Mashup 资源 www.floodlevels.com 得以发掘，那么剩下的只需要在客户资料（电子表格数据源）和该 Mashup 之上构建异质数据源集成的解决方案。另一方面，如果在资源发掘阶段无法找到或者实际并不存在可直接利用的 Mashup，那么用户只有考虑从现有的资源出发，着手从最底层开始构思多个异质数据源之间的 Mashup 逻辑，表达自己的需求。

● 应用构建

合理的 Mashup 逻辑从构思到实现，一个最基本的问题就是如何在一个用户友好的环境下，实现各个逻辑模块的组装（代替传统的编程）。于是，良好应用平台的构建是当务之急。首先必须考虑到对于 Mashup 企业级应用来说，它的使用者是最普通的用户。这样的用户既不是一个 Java script 专家，也不懂 PHP/Java/Ruby 这些编程语言。其次，我们还必须考虑到即使用户在资源发掘阶段能够找到了一个现存可利用的 Mashup 应用，但是如果该站点对外不提供 API，那么这种情况甚至还要求用户自己构建屏幕抓取程序来处理该站点，这是不现实的。也就是说，广大最普通的需求者亟需一个良好的平台去构建自己的应用。

[4]中对如何方便构建 Mashup 进行了一些基本分析。如图 3 所示，构建 Mashup 最直接的方法是用编程语言自己动手编写一些过程性的代码（Procedure Code），这种方法对使用者技术要求很高，哪怕一个细小的读取操作都要由使用者自己来指定实现细节。第二种方法是采用类似于 XQuery 的声明性查询语言（Declarative Queries），这种方法实现起来更为简单，至少使用者只需要遵守查询语言的格式，指明操作步骤，而不用关心每一个细节的具体实现；然而对于广大最普通的需求者来说，我们不能做出他们熟悉计算机语言的假设。实际生活中，往往图形化的开发平台才是最能虏获人心的。

由此看出，构建 Mashup 应用所需要的平台应该至少应具有如下两个特点：容易使用，表达力强。实际中的 Mashup 应用可能涉及到的数据源种类繁多，形态各异，涵盖 public APIs, XML/RSS/Atom feeds, web services, HTML 等等，因此如何搭建一个能很好处理多种格式异

质数据源的平台是极其具有挑战性的。

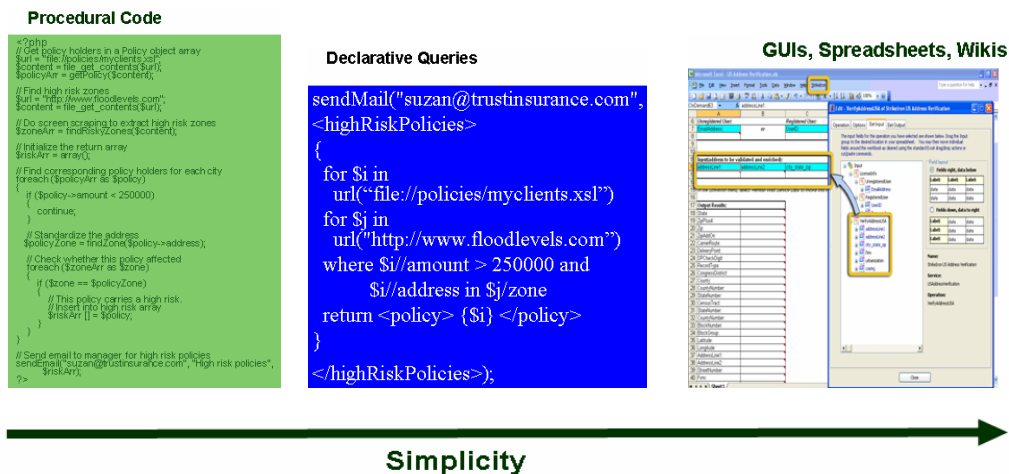


图 3: 构建自己的 Mashup

有的读者可能会问,帮助广大最普通的用户实现这种企业的即时应用是否还有更简单的方式呢?最理想的方式是,当用户在类似搜索引擎的环境下以自然语言的方式表达自己的整个需求,整个 Mashup 能够自动构建,就像搜索引擎一样直接将结果返回给用户,无须用户参与中间过程的构建。比如,例 1 中保险经纪通过搜索“flood levels for zipcodes 33101, 34106, etc.”而直接获得答案。这种智能化的人机交互方式首先需要自动发现和用户查询内容相关的,能帮助回答查询的若干资源(如图 2 中的若干网站);并且要在理解这些资源的逻辑的基础上,在资源间进行类似 join 的 Mashup 操作。[1]在处理 Deep Web 数据集成的过程中,自动选择合适的相关数据源进行查询,并通过数据源查询能力的组合来回答用户提出的查询。

<p>Source 1: Used cars for sale. Accepts as input a category or model of car, and optionally a price range and a year range. For each car that satisfies the conditions, gives model, year, price, and seller contact information.</p>
<p>Source 2: Luxury cars for sale. All cars in this database are priced above \$20,000 Accepts as input a category of car and an optional price range. For each car that satisfies the conditions, gives model, year, price, and seller contact information.</p>
<p>Source 3: Vintage cars for sale (cars manufactured before 1950). Accepts as input a model and an optional year range. Gives model, year, price, and seller contact information for qualifying cars.</p>
<p>Source 4: Motorcycles for sale. Accepts as input a model and an optional price range. Gives model, year, price, and seller contact information.</p>
<p>Source 5: Car reviews database. Contains reviews for cars manufactured after 1990. Accepts as input a model and a year. Output is a car review for that model and year.</p>

例如,在左图所示的 5 个数据源上,用户想查找 1992 年之后生产的运动车型的价格和评论。那么首先可以排除 Source 3 和 Source 4,因为这两个数据源不提供与用户查询相关的内容;其次我们需要组合 Source 1, Source 2, Source 5

的内容。Source 1 或者 Source 2 的输出与 Source 5 的输入在 model 和 year 字段上是可以做连接的,从而通过不同 Deep Web 数据源的横向连接,形成能帮助用户回答查询的统一视图。在这篇文章中,处理的对象相对来说比较单一,即 Deep Web 网站,而且假定对每个网站的查询能力都事先进行了分析,定义了明确的输入输出接口。然而,在 Mashup 应用程序中,我们面临的任务更加艰巨,如何理解各个类型数据源的处理能力,接口以及提供服务的范围成为了新的研究课题。

● 语义和非结构化

在解决 Mashup 数据集成的问题里,语义(Semantic)和对非结构化数据(Unstructured data)的处理是两大难题。语义方面举一个最简单的例子,在自己储存的一份客户资料里,客户的名字可能使用自己熟知的一些昵称或简称代替的,当某个 Mashup 应用需要处理到这



Dear Owen,

I write regarding our ACL paper's final submission (confirmation number 295).

I have recently carried out the upload of all three versions of the paper again, and I believe them all to be in proper format. If this is incorrect, I will be happy to make the appropriate modifications. In that event, I would be grateful if you would advise me what needs to be changed. The fastest way to reach me is at, 650-988-0674.

Thank you,
Phil Beineke

要在Email 中查 Beineke 的手机号

份资料并与另外一些信息在该字段上做连接的时候，机器是无法识别这些昵称或简称的。在关于语义的领域里，存在大量的研究工作。同时我们很欣喜地看到随着信息的高速增长，可用的标准化服务越来越多（将<male, female>, <M,F>, <1,2> 等表示同一语义的不同方式识别

来更多地被重视和使用，企

(1)

业中也逐渐将元数据的管理提上了日程。种种这些努力都在为逐步跨越语义这道鸿沟推波助澜。另一方面，人们总在试图在信息领域所有非结构化的数据上做工作，解析成为结构化的数据，这样机器就可以直接处理了。可是如何从非结构化的数据中抽取出结构化的信息是一个非常头疼的问题，尤其是 Mashup 企业级应用面临了不同于以往我们所考虑的数据抽取问题。举一个例子，当一个 Mashup 应用需要在一份客户资料和一封电子邮件之间在姓名和电话号码字段上进行连接操作，那么这时候的数据抽取问题就是全新的——如何从非结构化的电子邮件文本中抽取出<姓名，电话号码>这样的二元组。如图 4（1）所示，落款为 Beineke 的邮件中包含了一个电



Dear Owen,

One thing I forgot to add in my previous mail (re: confirmation Number 295).

If, for whatever reason you are unable to reach me, my co-author Shivakumar Vaithyanathan will be reachable at 410.555.1212.

Thank You
Phil Beineke

避免这种混淆引 入错误

(2)

图 4：从 Email 中抽取结构化信息

语义的理解和对领域知识的利用也变得越来越重要。从这个例子中，我们对搜索引擎未来的发展方向也可窥见一斑。用户或许可以只需要向搜索引擎提交简单的查询“Beineke phone”，搜索引擎就能将解析之后得到的<姓名，电话号码>二元组返回给用户，并且要保证返回结果的正确性，避免出现如图 4（2）所示的混淆情况。

话号码，那么这种情况下，我们能相对容易识别并抽取这样的姓名和电话号码二元组。但是如果出现如图 4（2）所示的情况，文中出现的电话号码并不是 Beineke 的，而是其伙伴的，那么对这样混淆情况的识别对新形势下的数据抽取提出了更高的要求。于是，在对非结构化数据进行解析的过程中，文本

5. 结束语

Mashup 是一种令人兴奋的交互式 Web 应用程序，它利用了从外部数据源检索到的内容来创建全新的创新服务。第一个关键词是“交互式”。实际生活中的 Mashup 应该是面向广大普通使用者的，简单易用表达力强的新型 Web 应用。用户应该能在一个交互式的用户友好的平台上进行各种异质数据源的“组装”，而不是以繁琐编程的方式构造 Mashup 应用程序。当然这样的平台必须是足够智能化的，它不仅需要帮助用户寻找现成可利用的资源，

还需要使得具体实现细节透明化，如对语义的理解和对非结构化数据的处理等。第二个关键词是“创新”。Mashup 应用将已有的来自多个数据源（public APIs, XML/RSS/Atom feeds, web services, HTML）的信息进行加工，融合和进一步利用，从而使之产生更大的价值。这有点

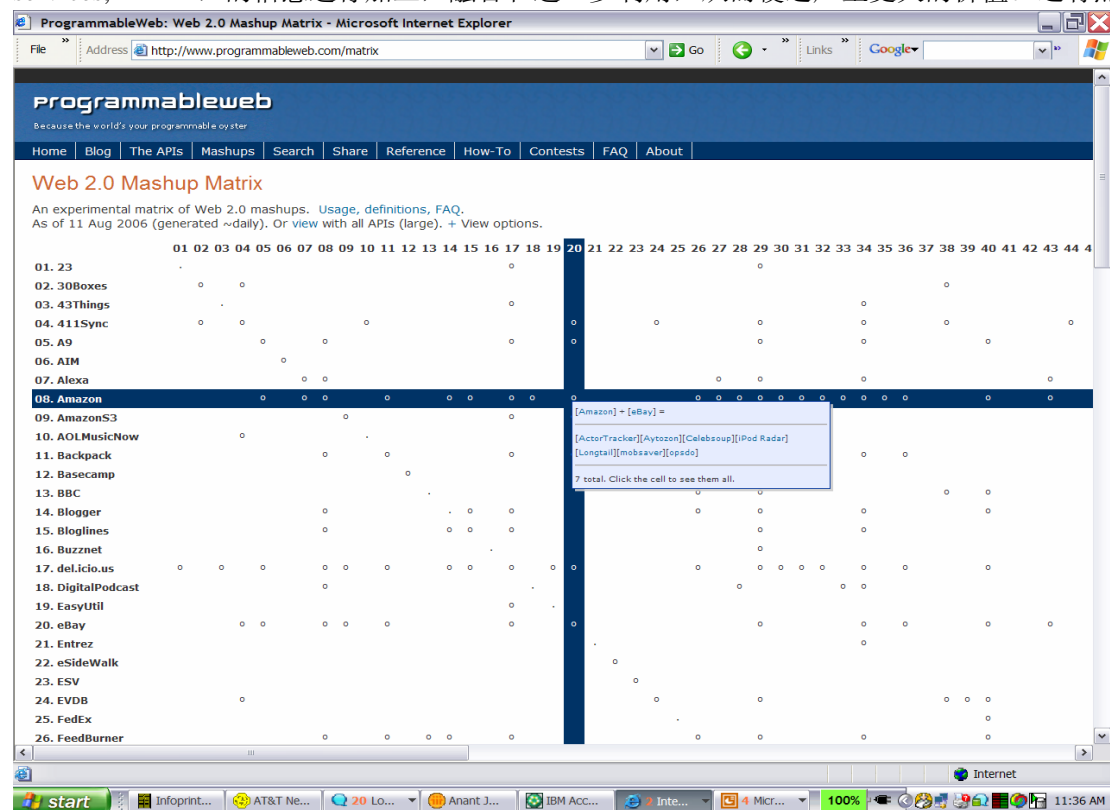


图 5: Mashup 矩阵

像做菜：把各种原材料按照某种方式烹饪在一起（当然不是简单的混合），形成一道美味。又比如把磨锋利的石头绑在木棍上，做成一个斧子，却不是石头和木棍两样东西功能的简单累加，也就是实现了“1+1>2”的突破。ProgrammableWeb.com 对目前 internet 上涌现的 Mashup 应用进行了概要性的统计和描述。如图 5 中 Mashup 矩阵所示，矩阵中每一行列交叉点都代表着已经存在这样的 Mashup，它在该行和该列所代表的信息源的基础上进行了组合和创新。随着越来越多的信息源开始对外公开了自己的 API，越来越多的有趣的 Mashup 应用也如雨后春笋般涌现，这个 Mashup 矩阵将不再稀疏。

参考文献

- [1] Alon Y. Levy, Anand Rajaraman, Joann J. Ordille: Querying Heterogeneous Information Sources Using Source Descriptions. VLDB 1996: 251-262
- [2] <http://www-128.ibm.com/developerworks/cn/xml/x-mashups.html> Mashups: Web 应用程序新成员 (Duane Merrill, 2006)
- [3] <http://www-128.ibm.com/developerworks/cn/db2/library/techarticles/dm-0602lurie/> DB2 和开放源代码: 在 Linux 上使用 Google Maps API、DB2/Informix 和 PHP 创建地图 (Marty Lurie 和 Aron Y. Lurie, developerWorks, 2006)
- [4] <http://aitrc.kaist.ac.kr/~vldb06/slides/K-1.ppt> Anant Jhingran: Enterprise Information Mashups: Integrating Information, Simply. VLDB 2006.