Weaving

设计文档

By JerryMouse

yankai
2010/6/1

目录

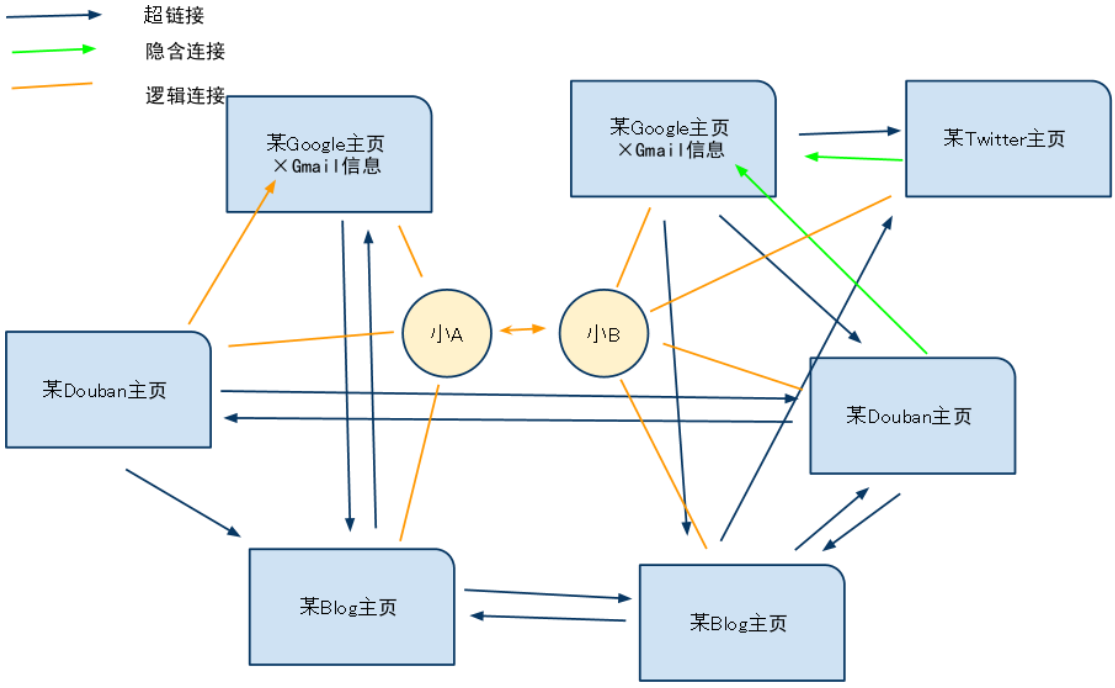
工作机理 2
互联网 3
整理 3
整体结构 4
Weaving-Web 4
Weaving-API 5
Weaving-person-digger 5
Weaving-website-extracer 5
weaving-repository 5
weaving-eye 6
数据模型 6
SNS 产品研究 6
SNS 产品案例分析 6
SNS 社区的三层产品架构； 6
模型 7
模块介绍 9
Weaving-Web 9
页面列表 9
Weaving-API 9
Extract API 9
Dig API 10
Weaving-person-digger 11
挑战 11
解决办法 11
计划中 11
Weaving-website-extracer 11
挑战 11
解决办法 11
设计 11
示例 Filter: 12
weaving-repository 12
weaving-eye 12
子项目 12
JSA4j 12
Jsa4j 子项目列表 12
Jsa4j-db-kv 13
get 14
put 14
GetFavicon 14

工作机理

Weaving 可以将杂乱无章的互联网捋成一张语义网，然后在通过语义网挖掘出“人”的信息。

互联网

- 杂乱无章的链接关系
- 隐含有语义信息

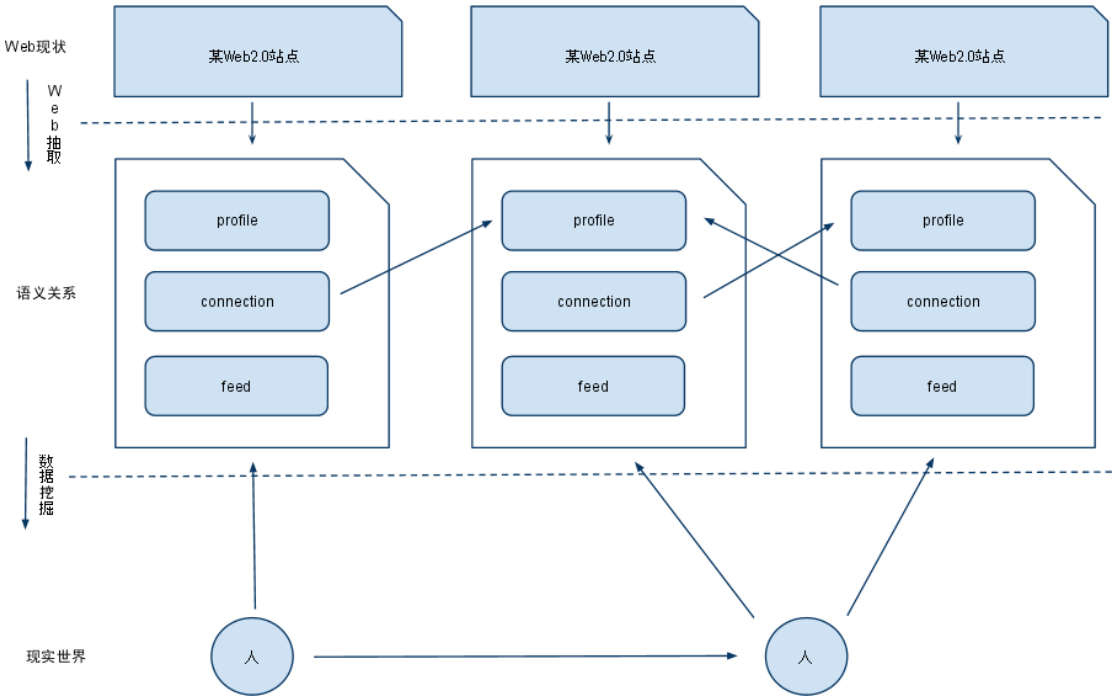


主要包含了四类信息：

- 页面本身的文本信息
- 页面直接的超链接
- 数张页面是同一个人创建的
- 创建页面的人之间的关联引起的页面之间的关联

整理

- 通过 Web 抽取，将互联网上的信息抽取为语义网
- 通过数据挖掘，从得到的语义网挖掘出“人”的信息



纵向关系

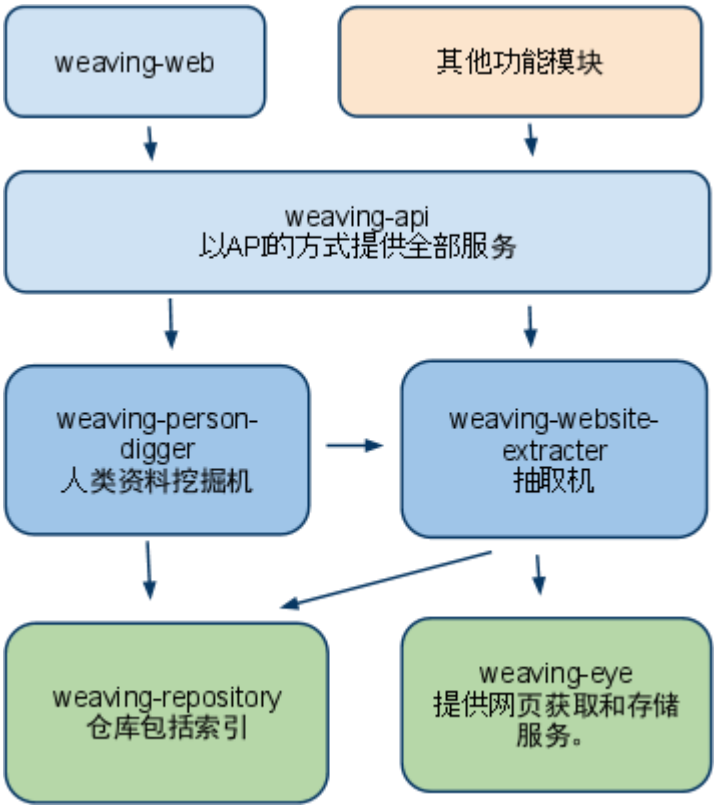
- 每一个事实的网页可以对应一张语义网页
- 一个人会和若干个语义网页关联，他们之间是创造和被创造的关系

横向关系

- 网页之间有超链接相连
- 人和人之间有人际关系相连
- 语义网之间有语义关系

整体结构

总共分为若干模块



Weaving-Web

Weaving 的 Web 前端，符合 Restful 的设计，有两个实现，一个是本地使用 Derby 为数据库。一个在 GAE 使用 BigTable 为数据库。

Weaving-API

Weaving 的核心 API，对外提供服务，有两个主要接口。
您可以在 http://weaving-gae.appspot.com/api 使用。

Extract API

根据一个 URL，挖出其相应的 WebSite 信息。该 WebSite 信息是结构化的。

Dig API

根据用户名，Email 或者 url，挖出其相应的 Person 信息。该 Person 由若干 website 聚合而成

Weaving-person-digger

从语义网挖掘出人的核心模块

Weaving-website-extracer

从互联网抽取出语义网的核心模块

weaving-repository

提供信息存储，索引，搜索服务。使用 Jsa4j 为底层

weaving-eye

网页获取，存储，索引。可以翻墙

数据模型

SNS 产品研究

我们对目前的 SNS 产品进行了一些研究，方便架构和模型的确立

SNS 产品案例分析

具体调查见《SNS 产品案例分析》。

调查结果

从推理的角度上讲，数据间有两类关系：
站内推理，如已知个人 ID，推出个人主页
站间推理，如推理出两个服务是同一个人的。

此外还有一些推理是绝对可信的，比如存在 Email 或者链接的信息，一些则是不那么可信，比如 仅仅是真实姓名相同就加以判定是同一个人。

如果一个 div 中出现 Friend,友等字样，其下链接是其友人的站内链接。

一个 Profile 中出现的同站链接多半为好友或 Follow 联机，其异站链接多半为自己的其他博客。

SNS 社区的三层产品架构；

从概念上，SNS 是一种新的在线生活方式；连 FACEBOOK 自己都说自己不是一个网站，而是在创造一种新的网络生活方式；这个理念和腾讯的“在线社区产业”是相通；

从外部一些文章可看到，腾讯形成了在线社区 3C 产业链，分为三层，从下到上分别是用户(Customer)，社区(Community)，内容(Content)；这是腾讯的创造性贡献；

其实，具体到一个 SNS 社区产品模型，从下到上也分为三层：

底层，Profile；用户的属性描述及行为画像；

比如用户的社会属性，姓名，性别，年龄，职业等；还包括用户的爱好，服务使用倾向等推导属性；
这相当于社区的“地基”，这里有几种细分：

一类是**用户的直接属性**；
表现为用户可以通过直接引导填写的信息；如姓名，年龄，性别，职业，毕 业年份等

基本社会属性；看到所有的 SNS 都在引导用户填写，甚至采用一些激励措施；

二类是**用户在社区中生存所获得的社区属性**；
比如成长等级，称号，虚拟职务，角色等；

三类是**用户的隐藏的扩展属性**；
即系统通过对用户在各类社区长久活动留下痕迹的智能挖掘 与分析，所形成的对用户有统计意义的商业偏好属性；比如用户 XX，是一个 30 岁左右，怀孕期的妈妈，对婴儿用品，化妆品有独特的潜在偏好；

一个不同完善程度的社区系统，对于一个用户信息的收集也是不同层次的；而所有的商业网站通过持久竞争，留下来最宝贵的核心竞争信息，就是对用户 个人信息的掌握能力了；

中间，Relation；用户群内部关系链；

在 WEB1.0 时代，每天浏览 SINA 的人 可能有 100 万，但他们虽然同在访问一个网站，同看一条新闻，但相互之间无法察觉，无法交流和沟通，这 100 万人中是孤立的，没有关系链；

随着 WEB2.0 元素的发展，网站经营者知道给每个访问的用户一个 ID，让他们相互可见，并提供他们相互联系，认识并熟知的工具和手段（比如站 内消息，相互访问首页）；

关系链，包括人与人的关系；人与群体的关系；群与群的关系；

具体表现为，好友关系（强关系链），关注追随关系（弱关系链），同好关系，（同爱好，粉丝圈）；同地域关系（同城）等；

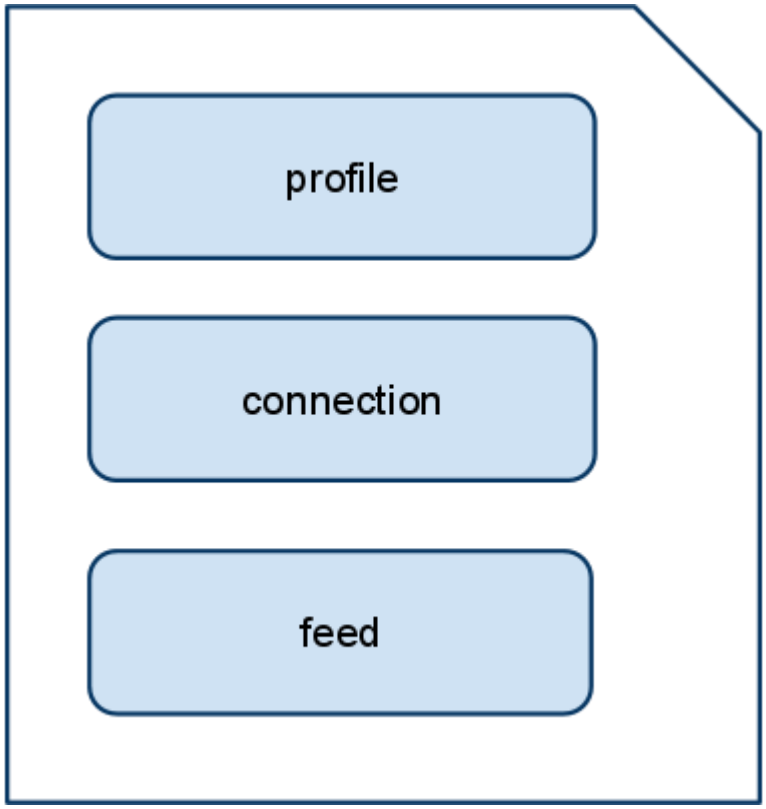
上层，内容(Content)；

内容(content)，包括两类，
一类，是网站经营者官方提供的资讯，图片，音乐，等浏览类的资源；
二类，是 UGC（User Generated Content），用户自创造自组织的内容；

可表现为，个人日志（Blog），相片，即时博客（如短文本 Qzone 心情，Twitter）；

内容，从表现形式及载体上从简单到丰富，从简单文本，短文本，到图片，音频，甚至个人视频，随着网络硬件条件的发展，内容的主流载体将更加 RICH 化；

模型



这个模块可以理解为 Web 抽取之后形成的带有语义的数据池。
如图一个社交网站由 3 部分组成：

- profile
- connection
- feed

profile 中，一般会有如下信息：

- 个人主页
- 用户名，一般可读
- id,一般不可读
- Emails
- 头像链接 s
- 基本信息(真实姓名,生日,性别,地点....)

connection 中一般有：

- Followed 的人到站内链接 s
- Follow 人的站内和站外链接 s
- 朋友的站内链接 s
- 自己的站外链接 s

其中 Follow 和朋友之间的不同点就是 follow 不需要别人同意，而 friend 必须被同意，同时 friend 必然是双向的

feed 中包括

- 信息源地址

Feed 引擎：
能提供 Feed 聚合业务,类似 friendfeed。

模块介绍

Weaving-Web

Weaving 的 Web 前端，符合 Restful 的设计，有两个实现，一个是本地使用 Derby 为数据库。一个 在 GAE 使用 BigTable 为数据库。使用技术为 Spring mvc+velocity。

页面列表

- / 主页
- /search 搜索页面
- /about 介绍页面

- /api api 介绍页面
- /api/extract 抽取 API
- /api/dig 挖掘 API

- /person/kaiyannju 个人页面

Weaving-API

Weaving 的核心 API，对外提供服务，有两个主要接口。可以提供给其他用户使用。您可以在 http://weaving-gae.appspot.com/api 访问。

Extract API

介绍

根据一个 URL，挖出其相应的 WebSite 信息。该 WebSite 信息是结构化的。

参数

http://weaving-gae.appspot.com/api/extract？q={url}

示例

分 析 http://www.google.com/profiles/KaiYanNju

返回

JSON 数据。数据模型为

- profile
  - id
  - host
  - url
  - username
  - avatarLinks
  - emails
  - personInfomations
- connection
  - followedLinks
  - followLinks
  - friendLinks
  - selfLinks
- feed
  - feedLinks

其中复数为字符串数组。

Dig API

介绍

根据用户名，Email 或者 url，挖出其相应的 Person 信息。该 Person 由若干 website 聚合而成

参数

http://weaving-gae.appspot.com/api/dig？q={username,email or url}

示例

挖 掘 http://www.google.com/profiles/KaiYanNju

返回

JSON 数据。数据模型为

- key
- websites
  - profile
    - id
    - host
    - url
    - username
    - avatarLinks
    - emails
    - personInfomations
  - connection
    - followedLinks

        - followLinks
        - friendLinks
        - selfLinks
    - feed
        - feedLinks

其中复数为数组。

Weaving-person-digger

从语义网挖掘出人的核心模块

挑战

语义网本身不准确。有假信息和信息不足。从不准确和不完整的信息源中，挖掘出相当可靠的信息是很困难的。
语义网太大，分析困难

解决办法

信息分级。将信息利用可靠性分级。有的信息天生很准确，如 Email。有的网站被解析的很准确，而有的不准。
从语义网提取出局部进行运算。先利用索引和推理，选出候选网站，减小问题域，再进行运算。
Agent(滚雪球的判别机器人) 。利用 Agent 技术，时间监视“人”单位，以助于发现新的关联或者剔除旧的关系。

计划中

利用机器学习来处理分级的问题。

Weaving-website-extracer

从 互联网抽取出语义网的核心模块

挑战

各大网站各不相同。页面千奇百怪。
独立博客
访问权限

解决办法

社交网站结构归纳，模型的确立
脚本引擎。三行搞定一类网站。[示 例代码]
动态策略选择
使用现有 API(如 Facebook Graph API)

设计

使用 Filter-Pipe 结构。
将线索逐渐经过 Filter 而丰富，同时管理策略，决定下一个 Filer 和什么时候结束。

利用 Groovy 脚本实现的 Filter 实例。极大的方便的一个 Filter 的建立。

示例 Filter:

LinkFilter.
通过判别 Filter 中的 Link 模式和 Link 是同站还是异站带，推测这个 Link 的语义
XENFilter
直接通过语义网标准来确定 Link 的语义
GoogleGroovyFilter
专门为 Google 定制的 Filter，只对 Google 有效。

weaving-repository

提 供信息存储，索引，搜索服务。使用 Jsa4j 为底层
由于 Jsa4j 已经封装了复杂的底层，这个模块基本用于 OR Mapping。

weaving-eye

网页获取，存储，索引。可以翻墙。翻墙功能需要使用代理。

子项目

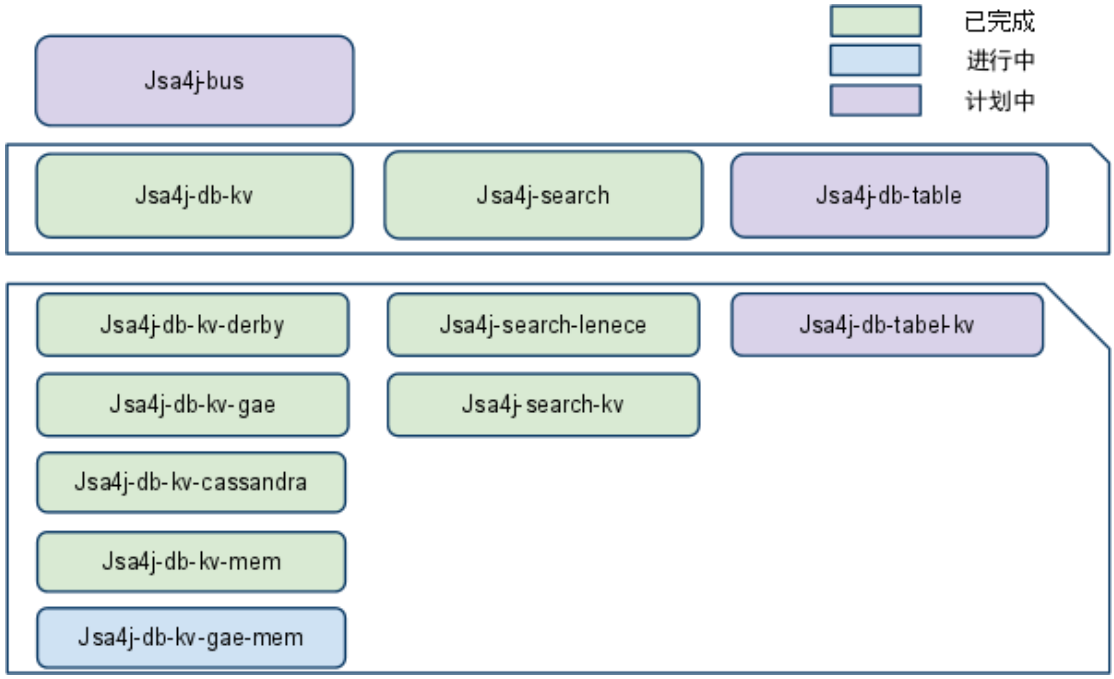
JSA4j

jsa4j 是 Jerrymouse Storage API for Java 的简称。是 JerryMouse 小组开发的通用数据底层，可以架设在单机或者 Gae 环境之下。脱胎于 CommonCloud 项目，由于 CommonCloud 过 于复杂，缺乏可用性。所以开发了他的简化版 Jsa4J。Jsa4J 的目标是可用和简洁。

Jsa4J 给技术力量薄弱的团队，提供使用 Nosql 数据库的可能性。可以用 Jsa4j-db-kv-derby 开发，运行在 sa4j-db-kv-gae 上。也简化了数据库的开发。

地址 http://code.google.com/p/jsa4j/。

Jsa4j 子项目列表



Jsa4j-db-kv 提供 KeyValue 数据库接口。有一个 Derby 和一个 Gae 实现。还有用于缓存的支持

Jsa4j-vfs 提供虚拟文件系统支持。

Jsa4j-db-table 表结构的数据库支持

Jsa4j-search 提供全文搜索支持。

Jsa4j-bus 建立在分布式缓存上的通讯总线

Jsa4j-db-kv

简介

随着 NoSql 运动，新奇的数据库层出不穷，提供 了各种丰富的接口。这些接口丰富在两个方面：

- 事务处理
- 数据结构

Jsa4j- db-kv 没有“事务处理”和“数据结构”的概念，极大的方便了数据库开发。

关于事务

不管是 ACID 还是 BASE，都是事务处理方式。Jsa4j-db-kv 没有事务的概念，默认大于配置， 认为存操作需要事务，取操作不需要。认为数据库写入永远是成功的。具体是不是真的能成功，应该由另一套系统来管理。

对于比较可靠的列存数据库，和不怎么可靠的类似 Cache 的数据库都有支持。

关于数据结构

数据结构方面有关系性 数据库，列存(BigTable like),文档数据库，图数据库和 Key Value 之分。其中 Key Value 是最简单的，可以由其他类型的数据库实现。同时提供一个索引工具和搜索工具，满足在数据索引上的需要。

接口

Jsa4j-db-kv 只是一个 KV 数据库接口。
接口本身非常简洁：只有两个方法：

String get(String key)

String put(String key, String value)

详细

get

`String` **`get`**`(String key)` 取操作
参数：`key` - 键 **返回：**值，理论上是一个 json 对象。当然看你存入的是什么了

put

`String` **`put`**`(String key, String value)` 存操作。

- 如果 Key 为 null,value 也为 null,则不操作
- 如果 key 为 null,value 不为 null,**不操作而不是执行 insert 操作**
- 如果 key 不为 null,且在数据库中存在对应的 value, value 也为 null，执行 delete 操作
- 如果 key 不为 null,且在数据库中存在对应的 value, value 不为 null，执行 update 操作
- 如果 key 不为 null,但在数据库中不存在对应的 value,,value 为 null,则不操作
- 如果 key 不为 null,但在数据库中不存在对应的 value,,value 不为 null,执行 inerst 操作

注意点：DB 提供的是数据库的底层操作。不提供主键生成机制。
参数：`key` - 如果是 insert 操作，key 中必须含有类型信息，否则结果不可预知 `value` - 一个 JSON 对象，也可以是别的 **返回：**刚刚操作的那个对象的 key

满足这个接口开发，就可以使系统有很好的可迁移性。

GetFavicon

网址：http://favicongetter.appspot.com/

- 可以通过域名来获得一个网站的图标，简单实用。

- 基于 Google CDN。可以获得良好的速度。
- 附带一个基于 CSS3 的图标编辑器。提供 API 服务。