# Design Analysis for Web Analytics Site

Andrew Song (andrew90)

## 1. Overview

The Web analytics site is intended to help the owners to track visit and duration counts to their own websites. There are two goals to this project: 1) Familiarizing with web development and its languages such as Ruby on Rails, Javascript, HTML, and heroku. 2) Creating light-weight, minimum viable application for Web Analytics. By inserting the JavaScript snippet provided in the Analytics site, the owner should be able to begin tracking visits without extra set-ups.

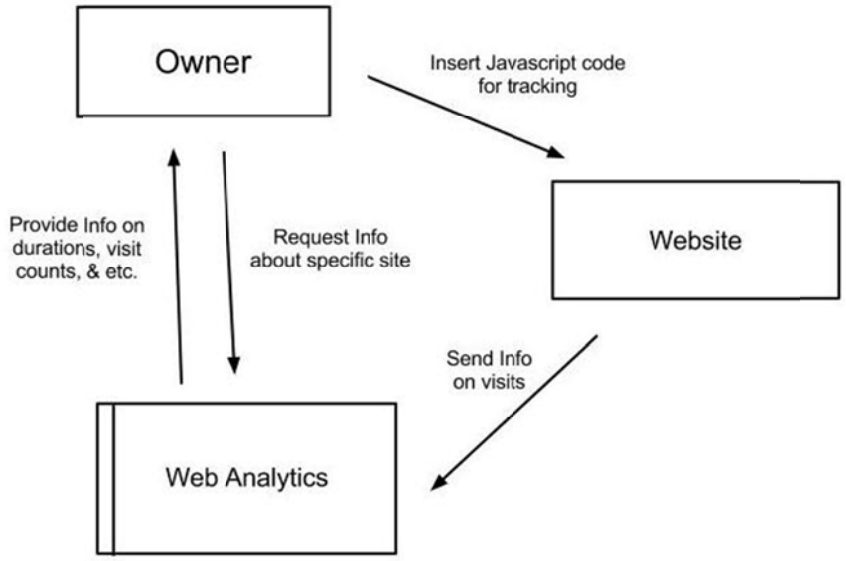


**Fig 1. The Owner and The Website are external components to the Web Analytics app. Each edge represents the event and the arrow represents the flow of data. Notice that HTTP requests are not included in this diagram.**

The above figure demonstrates the conceptual diagram of this application. Once the Javascript snippet is inserted into a specific website, the website will send information on visit when a there is a visit. Web analytics will collect and accumulate the data in its database; the owner can find the visiting information about the specific website on the Web Analytics site.

## 2. Concepts

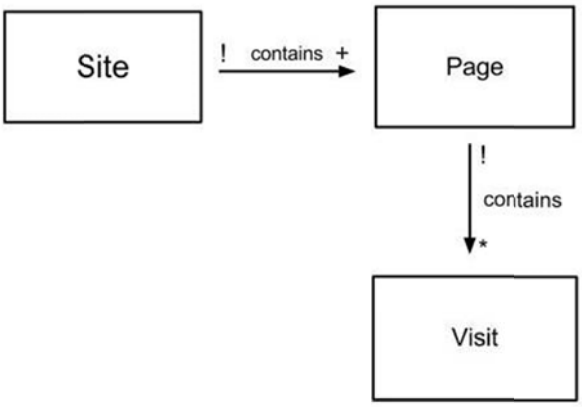


Fig 2. Object Model for Web Analytics

The models in the Web Analytics are **Site**, **Page**, and **Visit**. Site contains one or more page objects. Site governs the whole website, whereas page objects govern the subpages to that host site. For example, http://www.google.com would be governed by a Site object, whereas http://www.google.com/mail, and http://www.google.com/docs are governed by Page objects. A page object contains visit objects. Each visit object will have duration, URL, enter_time, and leave_time as attributes. The duration and the visit counts of the site object would be the accumulated sum of those of its subpages. Moreover, the duration and the visit counts of the page object would be the accumulated sum of its visit objects.

## 3. Challenges

- **Copying the Javascript Snippet**: The user shouldn't be able to assign the id manually. Only when the user uses the Javascript Snippet with the generated id, then the Web Analytics will work. There might be collisions between users if the user is able to self-assign the id.

- **Tracking visit duration**: The problem of tracking the visit duration was different from that of tracking the visit count. Whereas the visit count depended solely on the fact that the user "entered" the website, the duration depended on both the user "entering" and "leaving" the website. Here are two possible implementations I have contemplated over:

1) When the user enters and leaves the site, send the POST request along with timestamp parameter (using window.onload & window.onunload). The Web Analytics will take in the difference between the two parameters to calculate the duration.

2) When the user enters, the client browser will assign a current timestamp to a variable (enter_time). When the user leaves, the browser will send the POST request with enter_time variable and the timestamp when leaving the site. The Web Analytics will take in the difference between the two parameters to calculate the duration.

The difference between the two would be whether to send two POST requests for single visit, or just send a single POST request at the end of a single visit.

I opted to go with the second implementation. The first implementation, although intuitively more optimal, imposes a problem when multiple user tries to access a website asynchronously. Without the concept of sessions, it is hard for the server to keep track of unfinished visits (the user hasn't left the page). The server will have to identify each POST request of the user leaving the website and assign "leave_time" to proper visit objects. Second implementation doesn't need states or sessions in that once the visit object is created upon the POST request, the object is final and no further changes are needed to be made.

- **Security Concern**: The concern could be simplified to this: "Should owner A be able to see visit tracking information of other sites that he doesn't own?" The obvious answer would be no. However without the knowledge of security implementation and authentication, such features are yet to be implemented. At this point, a user can see the data of every page the Web Analytics is keeping track of.

**- Separating Site, Page, and Visit model:** Page has many visits, and a visit belongs to a page. Page has URL, visit_counts, and visit_duration as its attributes; Visit also has url and duration. Then why do we need to separate them at the first place? Wouldn't having just single site model suffice?

Having Page and Visit separated provides more information about the visits. Page model just tells you about total visit counts and visit durations. However, the owner might want to check how

long each visit lasted, and exactly at what time the visits have taken place. Although not implemented in this project, further details for the visit object can be implemented; IP address, regions, and etc. Having separate Visit model opens up the possibility for more in-depth analysis.

The tradeoff for such implementation would be that the visit object has to be created for each single visit, which might prove to be expensive.

Same logic applies to separating Site and Page models. Whereas the Site model only keeps track of the website as a whole, the Page model contains more specific information about each page.

**- Immutability:** Although the scaffolding provides CRUD operations, for the purpose of the project I removed all the operations except for Create and Read. The owner should only be able to browse the data, or create new tracking id to insert into a website. Updating and deleting the visit objects should be out of owners' concerns.

**- Code :** Still in the step of learning Ruby on Rails, Javascript, and AJAX, there have been several obstacles and decisions in actualizing the project.

1) Why not initialize the variables in visit method of sites_controller for each model, instead of creating setValues() method? => There seemed to be no concrete equivalent form of JAVA or C constructors in Ruby on Rails. There were lots of ways, and there were controversies on whether each of them worked properly. Moreover, Mas-assigning are thought to vulnerable to security measures. That's why I chose to go with creating new setter methods.

2) When initializing URL for each model, why change to [:sitePath] from [:siteURL] for Page and Visit methods? => The site doesn't need to know the whole URL of the origin of the request. It just needs to know the host. Page and Visit only have to know the sub-path because they are already associated with the Site object. Moreover, if the whole URL was used for Page, it would have taken longer time to search for the corresponding page in the hash, since the whole URL is longer than just the subpath.