6.170 Project 1.2
Design Document
Carolyn Zhang
carolynz@mit.edu

Overview

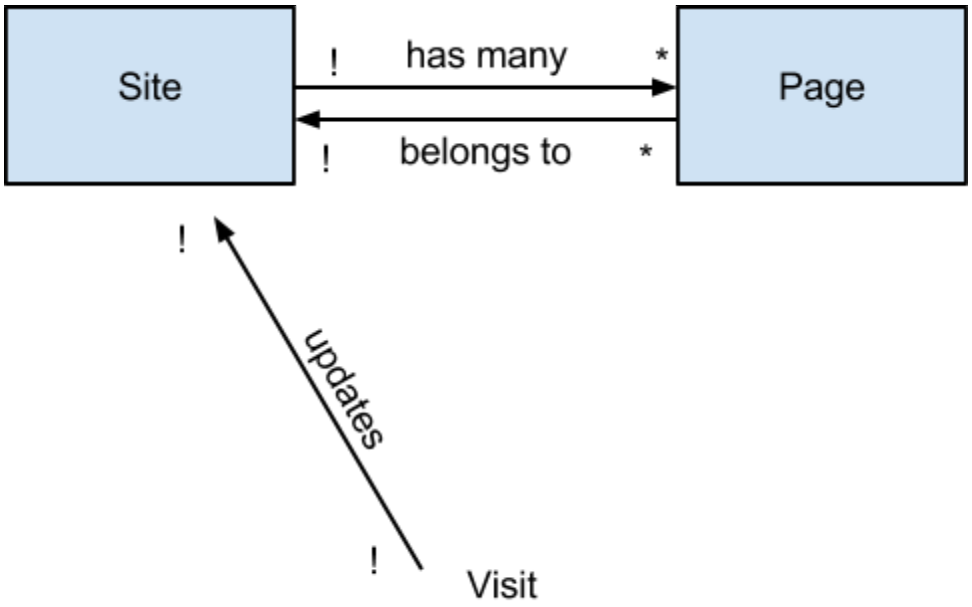
Purpose and Goals

In this project, I created a simple web analytics service that registers individual page visits on sites that the service is tracking. Currently, many web analytics services are overly complicated for the average user who just wants to see how many people are visiting their websites, and for how long. The simplest services that currently exist are:

1. hit counters (where a website will have a small widget that shows all visitors the total visitor count so far)
2. Google Analytics

Hit counters are too public, too limited in functionality, and may make the user seem narcissistic, whereas Google Analytics may have too much data and jargon that the average user doesn't need. This project aims to serve as a middle ground between the two options.

Context Diagram



Concepts

Key Concepts

Upon receiving an AJAX visits request for a page, the model's site controller searches for the appropriate site and page, and updates their stats as necessary.

Object Model

Sites have attributes that store its average duration, ID, name (determined by user entry), and total visits. Pages have attributes that store its average duration, parent site ID, URL, and total visits.

Challenges

Design Challenges

- **Receiving requests with unregistered site IDs**
  Options available:
    a. track the new site automatically
    b. deny the request, and force users to register each site that they intend to track
  The Javascript snippet provided to users who register a new site includes the correct site ID. Therefore, any requests we receive with an unregistered site ID indicates that someone has most likely manipulated the request (though there really isn't much user motivation to do so). Denying all requests with unregistered IDs is a simple way to prevent such manipulation.
- **Receiving requests with unregistered page URLs, given that the site ID is valid**
  Options available:
    a. track the new page automatically
    b. deny the request, and force users to register each page that they intend to track
  It is very likely that a user may frequently update their site with new pages (for example, in a blog), and it would be a hassle for them to manually register each new page's URL with the analytics server. Therefore, the server should automatically track new pages, given that the site ID is valid.
- **Calculating the visit duration**
  Options available:
    a. calculate all times server-side
    b. calculate all times client-side (in the JS snippet)
    c. calculate the one of the times server-side, and the other time client-side
  Calculating server-side ensures that the visit duration is not vulnerable to manipulation and that people can't fake extremely long visits. However, even if we do handle all times server-side, a user could just leave a page open for an incredibly long amount of time to achieve the same effect. That being said, there is little motivation for users to manipulate

visit times. Handling times client-side, though vulnerable to manipulation, is easier to implement (simply pass the duration as a parameter in the URL). Handling one time from server-side and one time from client-side is messy and unreliable, so it should definitely not be used. Therefore, I chose option B**.**

- **Registering page and site visits**
  Options available:
    a. register a page visit in the pages_controller by updating the page's visits and average duration values, then calculate parent site stats based on the page's updated stats
    b. register a page visit in the sites_controller by updating the site's visits and duration values, then call a page's update method

  While option A may seem less redundant than option B, the parent site calculations rely on the page update methods working properly. After attempting to implement option A, I found that it was difficult to figure out what was working properly or not because the site stats updater was dependent on the page stats updater. Option B takes a few more lines of code, but having two separate methods update site and page stats made it easier for me to see what holes I had in my logic.
- **Representing the notion of a visit**
  Options available:
    a. Create a Visits object that belongs_to a Page object (Page has_many Visits)
    b. Create *avgduration* (average duration) and *totalvisits* (total visits) attributes of a Page and Site, which get updated upon each visit

  Option A is attractive because it allows greater flexibility in what information we can store about a visit. We could store all prior visits or include visit attributes like visitor location. Indeed, Option A would be preferable in designing a system that is intended to provide a robust analytics solution for users. However, the goal of this analytics service is to provide barebones stats about average visit duration and total visits. Since we don't intend to scale the operation, simply storing and updating a Page/Site's average duration and total visits attributes is enough for this system.