BUILDING AN OPEN SOURCE SOCIAL MEDIA AGGREGATION TIMELINE

A Capstone Project Submitted to Southern Utah University

in partial fulfillment of the requirements for the degree of

Master of Arts in Professional Communication

January 2014

By

Justin Aiken

Capstone Committee:

Jon Smith, Ph.D., Committee Chair

Lance Jackson, M.A.

Lindsey Harvell, Ph.D.
The undersigned, appointed by the dean of the Humanities and Social Science, have examined the thesis entitled

Building An Open Source Social Media Aggregation Timeline

Presented by Justin Aiken

a candidate for the degree of Master of Arts in Professional Communication

and hereby that, in our opinion, is worthy of acceptance.

__________________________
Chair of Committee – Jon Smith, Ph.D.

__________________________
Committee Member – Lindsey Harvell, Ph.D.

__________________________
Committee Member – Lance Jackson, M.A.
Abstract

An ever growing number of web services and social media networks allow one to post a lot of content in to many different websites and services; however, there is no simple way to chronologically aggregate the content – especially not in a space controlled by the user. Relying on a site or service to present the user’s data through their interface allows for too much noise and muddies the user’s message. In response, this project built an open source, expandable timeline that pulls in a user's content from various places on the web, and hosts it on the user's own personal timeline, allowing them to control their self-presentation to the world through the Web.

Keywords: social media, open source, uses and gratifications theory, self-presentation
# Table of Contents

**APPROVAL PAGE**.................................................................................................................................... 2

Abstract...................................................................................................................................................... 3

Introduction................................................................................................................................................6

Literature Review.......................................................................................................................................7

  Social Media........................................................................................................................................ 7

  Conventional Websites.........................................................................................................................7

  Facebook.............................................................................................................................................. 7

  Twitter, Google Plus, LinkedIn, and Other Social Media Sites........................................................... 8

Uses and Gratification..........................................................................................................................9

UGT and the Modern Web.......................................................................................................................10

Self Presentation................................................................................................................................ 10

Online Self Presentation.......................................................................................................................11

Facebook's Timeline...........................................................................................................................12

Rationale.................................................................................................................................................. 12

Visibility.............................................................................................................................................13

Noise.................................................................................................................................................. 13

Aggregation........................................................................................................................................14

Aggregation through Existing Social Media......................................................................................15

Customized or Manual Aggregation..................................................................................................15

Existing Solutions...............................................................................................................................16

Justification........................................................................................................................................ 16

Method.....................................................................................................................................................17
Building an open source social media aggregation timeline

The Internet provides a dizzying array of different websites and services. Almost everything imaginable is available in an online representation. Many people use several different websites on a frequent basis. Internet users want to use a social network to check up on family members, update their online resume, save a new favorite recipe, find a new TV show to watch, upload photos they took, or any number of near-infinite possibilities. Some of the most common activities done on a daily basis include searching for information, communicating via email, social networking, following the news, or just passing time (Pew Internet, 2012).

Social media is one of the most common uses, with 73% of adults using at least one social media site, and a full 42% of adults are now using multiple social media sites (Duggan & Smith, 2013). Non textual content is also on the rise; 54% of internet users have posted – not merely viewed or used, but posted – photos or videos to an online site (Duggan, 2013). With so many avenues that an individual now has available to post their content to, and with more Internet users utilizing multiple avenues to communicate and post content, a new need has arisen to curate and aggregate this content.

For example, a hypothetical photographer posts their pictures on Instagram, writes about technical aspects of photography on a personal blog, and has an online resume page on LinkedIn. If they wanted share that information with potential clients, their choices would be either have to give potential clients all three links, or do a large amount of manual cross-posting. The limitations of these options are fully examined after a closer look at the literature, but neither are ideal.

Following research on Internet usage, social media, and the options available to a user to present their image online, this project consisted of writing a tool that aggregates content posted on various social media sites into a single timeline under the control of the user. The literature is reviewed
below, followed by closer examination of the limitations of existing methods for online self-presentation. Afterwards, the solution I build is detailed, and the work that went into developing the software is explained, along with limitations and future possibilities.

**Literature Review**

**Social Media**

One of the largest segments of internet use revolves around social networks, which can be defined as "a dedicated website or other application that enables users to communicate with each other by posting information, comments, messages, images, etc" (“Social Network”, 2013, def. 1). Some social networking sites focus the communication between users that have a connection. This connection can be between one user and another user, or between one user and many users. Still other social networking sites focus on content rather than connection. For example, Reddit allows anyone to post content or communication in any sub community chosen, and anyone viewing can vote or comment on the submission (Mieghem, 2011).

**Conventional Websites**

Before any social networks or blogging sites emerged, the option for posting content online was the traditional or conventional website. A personal website created the opportunity for anyone to be a mass communicator (Dominck, 2009). Often hosted for free as part of an internet service provider or in return for advertising space, the personal webpage provides an online 'home base', a central presentation of identity in cyberspace (Papacharissi, 2002). The typical personal webpage consists of static content; new information must be added manually if at all; for this reason, personal websites have become less common in the age of constantly updating social media (Zeldman, 2008).

**Facebook**

Facebook is the largest and most well known of the social networks (Duggan & Smith, 2013).
TIMELINE PROJECT

Founded in 2004, it has expanded to have well over one billion users – with 75% of US college students using Facebook (ibid., 2013), it's more likely anyone reading this paper uses Facebook than that they do not. In fact, Facebook's prevalence is so large, that outside Asia, Facebook is used by fully half of the world's internet users (Hackman, 2012).

Facebook initially provided each user with a profile page, direct messaging (an alternative to email), and a 'wall,' a type of personal message board they can use to share public messages, or have other users write messages on (Smith, 2009). Activity on the users' walls showed up in a news feed, letting a user see what action their friends have done recently, such as wall posts they've made or commented on, or changed aspects of their profile page (Pempak, Yermolayeva, Calvert, 2009). Over time, Facebook has added many other aspects – groups, pages, events, applications, etc, that also show up in the news feed.

Twitter, Google Plus, LinkedIn, and Other Social Media Sites

With the success of Facebook, competing/alternative social networking services have sprung up in market. Some of these focus on a niche; for example, LinkedIn is a 'Facebook for professionals’” (McKee, 2009), while Nextdoor is a Facebook clone for physical neighborhoods to interact online (Miller, 2013). Others are competitors offered by large companies seeking to leverage their existing market share into the social arena; Google closed down Google Buzz, their first foray into the social network market (Scott, 2012), and offer Google Plus as their new social network (Kaste, 2011).

Still, other sites don't have an interface similar to Facebook, but facilitate the sharing of content with others. Users of Twitter can post a message (called a 'tweet') limited to 140 characters to their publicly viewable timeline, which will also instantly appear to all of their 'followers' intermixed with all other tweets from people they follow (Signori, Segre, and Polgreen, 2011). Instagram and Flickr are two services that let someone easily share photos online and are in constant competition (Smith, 2013).
Vine has a rapidly growing marketshare for uploading and sharing video (Lunden, 2013). In short, there is no shortage of networks available to post and share content, whether it be text, photo, or video.

**Uses and Gratification**

Uses and Gratification Theory (UGT) examines the reasons people choose to consume media, and why they choose which media they do (Levy & Windahl, 1984). Rather than a passive image of the typical viewer, it examines the active role media consumers have (Levy & Windahl, 1985, p. 109). In speaking of the effects on those watching television, UGT suggests that "the term 'effect' is misleading because it suggests that television 'does something' to [the viewers]... Nothing can be further than the fact. It is the [viewers] who are most active in this relationship. It is they who use television rather than television that uses them" (Levy & Windahl, p. 1).

In the oft-cited 1973 paper (Katz, Blumler, & Gurevitch), which reviews the then-current state of UGT research, the authors examine the evolution of UGT. It begins by looking at unrelated studies examining such examples as why children are drawn to comics (Lyle, Parker, & Schramm, 1961) or why one might choose to listen to soap operas (Herzog, 1942) that formed the initial research into UGT. They then look at some of the studies that started to tie the disparate threads together, before looking at the state of UGT research at the time. Although the different strands of research were starting to come together, Katz et al (1973) still sought to see a "relevant theory of social and psychological needs" that went beyond a simple catalog and was a "clustering of groups of needs, a sorting out of different levels of need, and a specification of hypotheses linking particular needs with particular media gratifications" (p. 513). Since then, UGT research has indeed grown and expanded, evolving to keep up with cable, then satellite TV, then the Internet, then the rise of streaming video. An example of a paper examining applications of UGT to the Internet can be found by looking LaRose & Eastin's (2004) look into a new model of media attendance.
UGT and the Modern Web

A study looking at Twitter (Johnson & Yang, 2009) posits that social media allows users opportunities to fulfill motivations and gratifications never before possible. This new potential has seen Uses and Gratification Theory often applied to social networks and webpages. One such study (Raacke & Raacke, 2008) found that a large majority of college students are using social media sites for a large part of their day, and listed eleven separate reasons. A study looking at motivations behind creating personal webpages (Groner, Weibel, & Wissmath, 2010) found that the most common motivators were enjoyment, exercise (of web skills), self-portrayal, presentation of leisure activity, and job-related presentation. Another study examined motivations for Facebook use, and among the gratifications found, a common one was to make themselves look as attractive as possible, and convey a specific impression of themselves (Day, Dong, & Urista, 2009). They also noted that feedback users received from their wall posts caused some users to become addicted to checking Facebook for responses to the presented self (ibid, 2009). Another study looked at Facebook specifically to move past the obvious 'keeping in touch' uses and found that Facebook was an “important tool” for self-presentation, as well as a related use in social capital gamification (Joinson, 2008).

Self Presentation

Self presentation then is a common theme found in UGT research on new media, as well as one this project focused on. Self Presentation Theory explores how people present themselves to others. Goffman (1959) pioneered research in this area, looking at people's outward communication as an actor wearing a mask; thus we we carry out performances in all of our interactions. He states in his first book that “All the world is not, of course, a stage, but the crucial ways in which it isn't are not easy to specify” (Goffman, 1959). The performances we give are shaped by an objective to project a specific image to the audience – a carefully crafted image that the communicator desires to create, based on the
norms of the audience (Barnhart, 1994). Carrying out this performances is often unconscious, and undertaken no matter the mental state or confidence of the performer. Reasons for enacting this performance can range from actively seeking to achieve or change an outcome, such as presenting an image more likely to achieve monetary gain, or simply hide one's embarrassing habits (Goffman, 1959).

Goffman’s (1959) theory splits the human psyche into two stages – the front stage where we put performances on for the world, and the backstage area where we relax and be our true self. Impression management is the act of keeping these two areas stages separated, to avoid the dissonance that could be created if the buffer slips (ibid, 1959).

**Online Self Presentation**

The rise of social media has given new area to study Goffman's theories of self-presentation. Mehdizadeh (2010) found that narcissistic users with low self-esteem were more likely to spend more time online doing crafting their image through self-posting. A 2008 study looked at the identities constructed by Facebook users, finding they “show rather than tell” their identity and stress group or consumer identities over personal identities (Zhao, Grasmuck, & Martin). In looking at self-presentation on social media sites, Hogan (2010) found that a third stage is added to Goffman's front and back stage – web servers that decide which content you post goes to which audience. This role of the curator makes privacy controls very important to ensure user's postings – thoughts they may consider front stage for sharing with some audiences may very well be backstage material for another audience (Hogan, 2010).

Because communicating online gives a greater level of control over non-textual cues, users are able to carefully construct a controlled performances intended to present exactly the image they want to project (Papacharissi, 2009), making social networking sites a tool very suitable for careful self-
presentation. Some have went as far as to call it the 'ideal environment' for self-presentation (Donath and Boyd, 2004). Bullingham and Vasconcelos (2013) did a study examining social media participants, and found their participants were eager to recreate their offline persona self on the site, but a persona with facets edited; they stated that “this emphasizes the key premise in Goffman’s work that, when in ‘front stage’, people deliberately chose to project a given identity” (ibid, 2013, p. 101).

**Facebook's Timeline**

In 2011, Facebook introduced a new feature called “Timeline: The 'Story' of your Life” (Lessin, 2001), which at a first glance appears a perfect avenue to self-present with. Describing it as the “evolution of your profile”, Facebook described it grandly - “Imagine if there was an easy way to rediscover the things you shared, and collect all your best moments in a single place” (para. 6). Stripped of marketing speak, the actual patent describes a system or program for generating a social timeline, where multiple data items or events based on relationships are ordered based on time (Sittig and Zuckerberg, 2010).

In use, the timeline lets a user pick and choose what to highlight or hide from their profile – Facebook events such as Likes or Wall Posts, new 'life events' like moving or relationship changes; key to being useful to self-presentation, it also allows external applications. Content users have posted to other sites, such as photos to Instagram or videos to Vine, also can show up on this timeline. If someone thinks their Farmville victory is worthy of display on their personal Timeline, Facebook would even allow them star and display it.

**Rationale**

A study which examined users of a micro-blogging service similar to Twitter, looking closely at a dissonance between gratifications sought and obtained, found that the gratifications the users sought often went partially filled or – unfulfilled entirely (Wang and Zhu, 2012). It is therefore important to
examine the likely shortcomings found in the use of social media. This paper will focus specifically on the limitations encountered in social media for a user engaged in the self-presentation of their image online.

Visibility

One problem is how often and whether people actually see something a user posts to their friends. ‘Filter bubbles’ describes the algorithmic mean employed by websites to choose what users or messages to show to another user. In the best case, this is based on past history – in the worst case, based on whichever messages the server decides will be most profitable to be show advertisements around (Pariser, 2011). Facebook is therefore editing and curating content (Williams, 2013), and when they decide which of a user's posts to highlight or not, the algorithms are not taking a carefully presented identity into account!

Noise

In many academic writings on communication and communication theories, noise is a factor. Interesting enough, both technical writings on the physical transmission of analog or digital signals and communication theories often use the same types of terms to describe the level of noise interfering with the message; these terms coming from the Shannon-Weaver model, which is first described in a technical paper for Bell Systems (Shannon, 1998). This paper states that “signal may be perturbed by noise or distortion” (pg. 447), before expounding mathematical formulas describing how to calculate signal to noise ratio. Building on Shannon’s paper, Henkel (2012) discusses signal-to-noise ratios on social network sites. Describing Facebook, he states that a Facebook wall is overly polluted with irrelevant videos, articles a user's friends have read, obnoxious advertisements (sometimes hilariously ineffective and inapplicable for the user viewing), and communication between people the user does not care about. He goes on to say that despite the massive amount of noise, Facebook still has the
strongest signal compared to other competing networks, and concludes that in general, social media has an issue where either the signal is not strong enough, or there's just too much noise for the signal to get through.

Advertisements are one of the more annoying forms of noise in social media. Whether engaged in a realistic self-portrait or an idealized projection, while the user chooses carefully which content to display on their Facebook profile, advertisers and Facebook are choosing how to use that profile for targeted advertising (for an example of Facebook advertisement placement, see Appendix B). Describing a study on Facebook's advertising, Roberts (2010) says:

To test the effectiveness of Facebook’s advertisers’ micro-targeting method, Lessin (2008) did an experiment by creating his own Facebook advertisement. It was an ad targeted to his girlfriend, so he typed in her specific demographic (a Wall Street Journal Reporter, 25 years old, living in San Francisco, graduated from Harvard in 2006, majored in history, etc.) and was able to get the ad directly placed on her Facebook website. This test demonstrated advertisers’ ability to nano-target their market in a unique way not seen in traditional advertising (p 26).

While Facebook shows ads around a user's profile, LinkedIn is arguably worse. After creating an online resume and sending the link to a potential client or employer, when they view your profile they will see links to your direct competitors placed all around your resume. Even in a best case scenario where these distractions are absent, somebody viewing a user's social media page is not only seeing content the user wishes displayed, but the interface of the actual social media site hosting the content.

**Aggregation**
A larger void is created if somebody wishes to aggregate the content they post to various social media sites. As mentioned in the introduction, fully 42% of adult internet users utilize multiple social media sites (Duggan & Smith, 2013). They're sending photos to one site and posting thoughts on another, or posting video on one and research on another. What if a user of multiple sites wants one unified identity to present, where the perception of the identity is based on the content from multiple sources, instead of being colored by which site an observer sees a portion of their content on?

**Aggregation through Existing Social Media**

In 2009, Facebook opened up their service to allow external sites to hook into their system (Schofield, 2009). This allowed other sites to post their content into a user's Facebook newsfeed alongside activity their friends did within Facebook directly. This method does allow users to bring content into a central place – they can simply set up their Flickr or Vine account so photos or videos posted there all show up in their Facebook timeline. However, this places the users' information from not just Facebook, but other sites as well, into Facebook's “walled garden” - their data is now locked into Facebook (McCown, 2009), and the user is ever more tied into the whims of Facebook's needs when presenting their data for all their online activity. Appendix B shows how this might look.

**Customized or Manual Aggregation**

Someone wishing their content from multiple sources displayed interwovead together could always duplicate the process of posting. Every time they post content or a message on a website, they could post that same content or a link to the original content, into a central location – perhaps a personal website they fully control. Enterprising users may write their own program in order to aggregate it all onto their own website – but that is an effort requiring technical skills that either place it out of reach or not worth the time cost for the vast majority of the internet population.
Existing Solutions

One alternative is ifthisisthen that (https://ifttt.com/). This is a service that allows one to set up 'triggers' for various 'channels', and perform an action when the trigger occurs. It allows you to cross-post through setting up recipes – for example, you have it tweet whenever you blog, or blog whenever you tweet. It provides a very nice option, but it still doesn't let you go to a central service – you just copy content between various others. It also limited to mapping one service to another; it is not possible to one action to multiple sites. There are also sites that exist for site-specific crossposting; for example, Flickstagram (http://flickstagram.org/) imports Instagram photos to Flickr.

Other services work in the opposite direction from the aggregation needed – they set up an interface where you have a single control panel that lets you manage and schedule content postings to social networks. Sites such as Hootsuite (http://hootsuite.com) or Buffer (http://buffer.com) are used by businesses to manage their social media presence. On a smaller scale, Tweetdeck (http://tweetdeck.com) lets individuals manage multiple Twitter accounts as well as Facebook and Linkedin – only those 3 sites though.

Perhaps the service that comes closest to what is proposed here is Tint (http://tint.com) - they produce an aggregated timeline from various social media feeds. However, is a closed, proprietary solution. It is also non-expandable; there is no way to add a new service or site to it. It is not free, and is targeted solely at business users. The timeline they produced is hosted at tint.com – a user cannot integrate it into their existing website.

Justification

No solution exists that works with an existing personal (or business website), and pulls in content from an infinitely expandable number of social media sites and web services into a single
TIMELINE PROJECT

unified timeline. After initially integrating a solution like this to a website, and adding desired accounts, the user should be done; henceforth any activity carried out normally on the linked sites makes the data available to put on the ever updating timeline, keeping the content on the page fresh and dynamic. This content is presented free of advertisements, competing posts, or third party interfaces – just the user's data displayed in chronological format. Such a solution would keep a user's self-presentation in line with their intended image, and for professional use creates a living portfolio of their web activity.

Method

The project I carried out this semester was building just that solution; writing the software to create a free solution for a social media aggregation timeline. Seeking to create the exact answer to the problems listed above, I programmed a software solution to fulfill the needs experience by the hypothetical users. I called this software simply “my timeline” - see Figure 1 for a screenshot, or Appendix A for more screenshots, showing the same timeline with a handful of different themes or skins applied.
What It Is

The project is an engine built with Ruby on Rails. This is a framework that came about in 2005, and is optimized for programmer happiness and high productivity, letting the programmer write “beautiful code by favoring convention over configuration” (Hansson, 2009, para. 1). It’s rapid prototyping took off with the startup scene, and usage exploded; companies like Twitter, Hulu, Shopify, and Scribed pushed it's popularity. Currently over half a million websites are now built using Ruby on Rails (builtwith.com, 2013).

The timeline is written as an *engine* – any existing Ruby on Rails site can incorporate it by simply including a handful of lines of code. The timeline is written to both work with single-user use – such as an individual that already has a personal portfolio website, and desires to integrate an aggregated feed into that – and as multi-user capable, so a large site that has many user accounts can also give a timeline to each user. It could even be used with a skeletal Rails application to create a website that is nothing but the timeline for any number of users. Besides the timeline engine itself, I also built an application that worked just like this; the site existed to host the plugin. This served to both give me a way to demonstrate how the timeline engine works to people, and the source code of the demo app helps developers see how to incorporate timelines into their sites.

Open Source

Key to this project is the notion of “open source.” Open source can either mean that the source code used to create a piece of software is freely available and distributed with the project, or on a grander scale that it is built by “developers at many different locations and organizations sharing code to develop and refine programs” (Lerner and Tirole, 2002, p. 197). Much of the technology that powers the internet is built on open source technologies; web server software, operating systems, and frameworks are more often than not freely available community projects.
Github is the largest repository of open source work (Finley, 2013); many of the tools that used to create this project are already hosted there (including the central framework, Ruby on Rails.) Github makes it easy for interested users to clone the source code for a project, made modifications (such as bug fixes or feature enhancements), and push the changes back to the project owner for easy review and inclusion in future releases.

The source code for my timeline engine, the demonstration application, and all associated plugins are all published on Github, were it is free for anyone to download, copy, clone, or send bug fixes and feature enhancements. Besides the current state of the source code, the full history of all changes is also found there. The url for the timeline is http://www.github.com/JustinAiken/my_timeline and the url for the demonstration wrapper is http://www.github.com/JustinAiken/my_timeline-demo. The source code for these is also available in Appendix D. All the code is available under the MIT license (http://opensource.org/licenses/MIT), a more permissive open source license than the common GPL3 public license (2007) - basically it permits users to use the code or program however they like, so long as they don't sue the author.

**Expendable**

Central to this project is an expendable system. As mentioned earlier in this proposal, the amount of websites, services, and networks is dizzying and ever growing. For this reason, rather than hardcoding any particular service in, the core engine contains only what is needed for central functionality: aggregation, storage, display, authentication, and the ability to create posts outside of any third party site. To actually import or scrape data from other services, it has a flexible add-on system, so that each the import process for each service can have it's own separately maintained codebase. I built several plugins to demonstrate this.

First, I built a plugin for twitter. This connects the users timeline to their twitter account –
anything they tweet then appears on their timeline. Next, I added health graph, which is the service behind Runkeeper. This means that if a user goes for a run while their phone tracks their path, as soon as they finish and save the run, besides pushing the run to Runkeeper, they can also have it pop up on their timeline.

Finally, I built a plugin for Github, letting a user's open source contributions also post on the aggregated timeline. For programmers, this creates a truly useful living portfolio; open source contributors are highly employable: “The number one way of getting a job in any programming company right now is to have a GitHub account and show your work” (Begel, Bosch, and Storey, 2013, p. 52). Having a nice timeline of one's programming activity is more accessible and intuitive than trying to click through all projects they've contributed to on Github, and if this timeline also has some intelligent tweets on the subject aggregated in... it is easy to see how this can be used to create a better link to give out professionally than a reference to a Facebook or Twitter account!

Table 1: Addresses of the plugins created for the project

<table>
<thead>
<tr>
<th>Plugin</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Twitter</td>
<td><a href="https://www.github.com/JustinAiken/my_timeline-twitter">https://www.github.com/JustinAiken/my_timeline-twitter</a></td>
</tr>
<tr>
<td>Github</td>
<td><a href="https://www.github.com/JustinAiken/my_timeline-github">https://www.github.com/JustinAiken/my_timeline-github</a></td>
</tr>
<tr>
<td>Health Graph</td>
<td><a href="https://www.github.com/JustinAiken/my_timeline-health_graph">https://www.github.com/JustinAiken/my_timeline-health_graph</a></td>
</tr>
</tbody>
</table>

Outside of service expandability, the appearance of the timeline is also themeable. The engine renders the events as raw HTML with CSS attached – this allows a designer to construct CSS rules to make the timeline appear however they wish. The timeline can thus easily match an existing website it's integrated into, or a catchy theme can be created. Appendix A shows some screenshots of the timeline with different themes applied.

Technical Detail
This section is presented as an examination of some of the logic and concepts in the code; although tangential to the communication theory applications of the project, the technical discussion in this section presents a closer look at the work involved with the project. Also of note is Appendix B, where an ERD diagram shows the model scheme used by the core plugin and its interconnection modules. Most of the code should be fairly self-evident to other developers, since it follows standard Rails conventions, but there are a few places of note that I thought called for a more discussion.

First, in order to make the engine accessible to developers, I sought to make it as easy as possible to integrate into an existing application – I wanted no changes to the host application code to be necessary besides adding the engine/plugins to the Gemfile, tweaking the initializer, and mounting the timeline to the preferred route. To this end, I tried to use as much metaprogramming magic as possible to have the Timeline hook into the other parts of the application it needed access/modifications to, rather than requiring developers to put it in their own code. For example, the engine uses the Rails-settings gem to manage per-user settings, such as Oauth tokens and login info for the various sites; it also reprograms some parts of this library to work with the engine. Rather than requiring a developer to set Rails-settings up on their user model, I lazily initialized the engine, with a post-initialization hook that performed a class_eval on their chosen User model for them (Figure 2).

**Figure 2: Expanding the user model**

```ruby
RailsSettings::SettingObject.class_eval do
  self.table_name = "my_timeline_settings"
end

Mytimeline.config_object = ::RailsSettings::Configuration.new(Mytimeline.user_class) do |
  s, key :empty_placeholder
end

Mytimeline.user_class.class_eval do
  self.send :include, ::RailsSettings::Base
  self.send :extend, ::RailsSettings::Scopes
end

Mytimeline.config_object.key :twitter, defaults: { foo: "bar"}
Mytimeline.config_object.key :github, defaults: { foo: "bar"}
end unless Mytimeline.user_class == Mytimeline::UserStub
```
One other aspect calls for particular note. Most route-based mountable engines tend to use their own layouts – Spree, ActiveAdmin, etc. Since this engine is meant to be integrated more tightly with a public view, it uses the application layout; this means that path helpers to application routes would be broken when viewing a my_timeline route. To fix this, the following code was added to handle delegating missing path_to and link_to helpers back to main application:

```ruby
module MyTimeline
  module ApplicationHelper
    def method_missing(meth, *args, &block)
      ifmeth.to_s == '/path$1_url$1/
        if main_app.respond_to? meth
          main_app.send meth, *args
        else
          super
        end
      else
        super
      end
    end
  end
end
```

**Evaluation**

**Work Completed**

The final work consisted of 6,222 lines of code split across five separate projects (the core engine, the demo app, and the three plugins). There is also ample documentation included in each of the projects to assist other developers that want to work on the project. To help guard against bugs, the timeline project was hooked into a service called Travis-CI (http://travis-ci.org), which checks build status by building the project and running automated tests each time a change was made, to help ensure that a change did not break anything.

To get an objective quantifiable review of the code quality, I hooked each of the five projects up to a service called CodeClimate. Code Climate (http://www.codeclimate.org) is a site that runs automated scans through each of the objects, classes, and constructs used to build the software, and analyses the programming for best practices, the presence of patterns that result in disaster later, and
overly complex or unreadable code. It assigns a numeric grade based on GPA’s; 4.0 is the best possible score for code to achieve. Table 2 shows the scores for each of the projects in this work.

<table>
<thead>
<tr>
<th>Table 2: Code Climate GPA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project</td>
</tr>
<tr>
<td>Score</td>
</tr>
</tbody>
</table>

The core engine has currently been downloaded 550 times. On the source code page, several people have starred and subscribed to updates. So far, one outside contribution has been made to the code – a user submitted a change to the github plugin to improve the formatting of some documentation. At the time of writing this, one user appears to have started writing an adapter for Pinterest, but it is not yet complete.

**Conclusion**

For users of social media sites that want to present a specific image online, or at least shape their image – especially users of multiple social media sites – a new option is available that lets them maintain control of their own timeline. This is especially useful for those users that want a portfolio based site for professional reasons, that stays up to date with their activity automatically. As earlier quoted from Facebook: “Imagine if there was an easy way to rediscover the things you shared, and collect all your best moments in a single place” (Lessin, 2001, para 6).

**Limitations and Future Development**

This is a piece of software that has infinite possibilities for growth. The initial version was focused on filling a void found looking for something for aggregated self-presentation of a few chosen sites. The immediate next step is creating plugins for a plethora of services – I will probably do 5-10 more to cover sites I personally use – time will tell if others start creating plugins to serve their needs.
TIMELINE PROJECT

for sites I do not personally use. Past that though, I see the scope of the project expanding beyond pulling in events for display on a public timeline. I would like to grow it into an ever-increasing catalog of all personal data available online. This means expanding it beyond events, to scrape, aggregate, and display non-evented data in a meaningful way. This also means using it for uses outside public presentation. Currently it does have privacy options – a user could for example, hide a Tweet they didn't want to show on their timeline. However, it's primary purpose is public display, and that's how it's tooled. With future expansion, it could be used for self-quantification and personal tracking – a topic moving beyond the realm of communication.
References


Johnson, P. R., & Yang, S. (2009, August). Uses and gratifications of Twitter: An examination of user motives and satisfaction of Twitter use. In *Communication Technology Division of the annual convention of the Association for Education in Journalism and Mass Communication in Boston, MA*.


doi:10.1145/1555400.1555440

doi:10.1089/cyber.2009.0257


doi:10.1177/1461444808099577


Williams, I. (2013, December 5). The gender identify of facebook: is it a boy or a girl, and has that affected the mobilizing of its gender? Retrieved from https://onlineacademiccommunity.uvic.ca/iwanwilliams/2013/12/05/the-gender-identify-of-facebook-is-it-a-boy-or-a-girl-and-has-that-affected-the-mobilizing-of-its-gender/


## Appendix A: Various Screenshots of the Timeline

### Screenshot 1: Plain Theme

<table>
<thead>
<tr>
<th>Time</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>6:30 am</td>
<td>Ran 5.1 miles in 40:07</td>
</tr>
<tr>
<td>3:07 pm</td>
<td>Tweeted &quot;@Foo Hello! How are you doing today?&quot;</td>
</tr>
<tr>
<td>8:00 pm</td>
<td>Made a commit 34DFH3 to JustinAiken/timeline - Added a new test to cover fixed bug</td>
</tr>
</tbody>
</table>

**Sunday, September 15th**

<table>
<thead>
<tr>
<th>Time</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>5:30 am</td>
<td>Ran 15.1 miles in 80:02</td>
</tr>
<tr>
<td>1:07 pm</td>
<td>Tweeted &quot;@Someone Yes, I enjoyed it&quot;</td>
</tr>
<tr>
<td>2:37 pm</td>
<td>Tweeted &quot;Ate something for lunch&quot;</td>
</tr>
<tr>
<td>3:47 pm</td>
<td>Tweeted &quot;Just watched Breaking Bad&quot;</td>
</tr>
</tbody>
</table>

**Saturday, September 14th**

<table>
<thead>
<tr>
<th>Time</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>6:10 am</td>
<td>Ran 5.1 miles in 42:07</td>
</tr>
<tr>
<td>9:37 am</td>
<td>Made a commit 123BCA to ryanb/cancan - Fixed a bug</td>
</tr>
<tr>
<td>9:47 am</td>
<td>Made a commit 1GGBCA to ryanb/cancan - Fixed another bug</td>
</tr>
<tr>
<td>10:07 am</td>
<td>Made a commit 135F1A to ryanb/cancan - Fixed a bug the last bugfix introduced</td>
</tr>
<tr>
<td>11:07 am</td>
<td>Made a pull request #243 to ryanb/cancan - Various bug fixes</td>
</tr>
<tr>
<td>11:30 am</td>
<td>Tweeted &quot;Wow, fixed a big bug in cancan today!&quot;</td>
</tr>
</tbody>
</table>
Timeline

Sunday, September 15th

+ 6:30 am: Ran 5.1 miles in 40:07
+ 3:07 pm: Tweeted "@Foo Hello! How are you doing today?"
+ 5:00 pm: Made a commit 34DFH3 to JustinAiken/Timeline - Added a new test to cover fixed bug

Saturday, September 14th

+ 5:30 am: Ran 15.1 miles in 80:02
+ 1:07 pm: Tweeted "@Someone Yes, I enjoyed it"
+ 2:37 pm: Tweeted "Ate something for lunch"
+ 3:47 pm: Tweeted "Just watched Breaking Bad"

Friday, September 13th

+ 6:10 am: Ran 5.1 miles in 42:07
+ 9:37 am: Made a commit 123BCA to ryanb/cancan - Fixed a bug
+ 9:47 am: Made a commit 1GGBCA to ryanb/cancan - Fixed another bug
+ 10:07 am: Made a commit 135F1A to ryanb/cancan - Fixed a bug the last bugfix introduced
+ 11:07 am: Made a pull request #243 to ryanb/cancan - Various bugfixes
+ 11:30 am: Tweeted "Wow, fixed a big bug in cancan today!"
Screenshot 3: Green Theme

**Timeline**

**Sunday, September 15th**

+ 6:30 am: Ran 5.1 miles in 40:07

+ 3:07 pm: Tweeted "@Foo Hello! How are you doing today?"

+ 5:00 pm: Made a commit `34DFH3` to `justinAiken/timeline` - Added a new test to cover fixed bug

**Saturday, September 14th**

+ 5:30 am: Ran 15.1 miles in 80:02

+ 1:07 pm: Tweeted "@Someone Yes, I enjoyed it"

+ 2:37 pm: Tweeted "Ate something for lunch"

+ 3:47 pm: Tweeted "Just watched Breaking Bad"

**Friday, September 13th**

+ 6:10 am: Ran 5.1 miles in 42:07

+ 9:37 am: Made a commit `123B4C` to `ryanb/cancan` - Fixed a bug

+ 9:47 am: Made a commit `1GGBCA` to `ryanb/cancan` - Fixed another bug

+ 10:07 am: Made a commit `135F1A` to `ryanb/cancan` - Fixed a bug the last bugfix introduced

+ 11:07 am: Made a pull request #243 to `ryanb/cancan` - Various bugfixes

+ 11:30 am: Tweeted "Wow, fixed a big bug in cancans today!"
Appendix B: Examples of Noise Present in Facebook
Figure 4: Entity Relationship between Timeline models

Appendix C: ERD Diagram
APPENDIX D

The remainder of this document contains the source code for each of the my_timeline repositories. A handful of files have been removed from the primary plugin (namely some of the boilerplate for the dummy application used for tests) as well as the demo application (mostly the stylesheets used to create the theme). The other three plugins are presented in their entirety. These version are 0.1.0 of each – future updates will be available at:

- https://www.github.com/JustinAiken/my_timeline
- https://www.github.com/JustinAiken/my_timeline-demo
- https://www.github.com/JustinAiken/my_timeline-twitter
- https://www.github.com/JustinAiken/my_timeline-github
- https://www.github.com/JustinAiken/my_timeline-health_graph
# My Timeline

A social-media aggregation/display plugin

This is a Rails Engine to help pull in content from any number of social media sites, services, or websites.

The aggregated information is displayed in a unified timeline.

It is being developed with extensibility in mind - each service will have it's own plugin.

### What it looks like:

![Screenshot](doc/screenshot.png)

### Requirements:

- Ruby 1.9.3 or 2.x
- Rails 3.1.x or 3.2.x or 4.x
- Bootstrap (or bootstrap-named classes) - For the markup. Just stuff like `table.table-striped`, no structural markup from Bootstrap is needed
- Any standard ActiveRecord-compatible database should work

### Supported services:

- [Runkeeper](https://github.com/JustinAiken/my_timeline-health_graph)
- [Twitter](https://github.com/JustinAiken/my_timeline-twitter)
- [Github](https://github.com/JustinAiken/my_timeline-github)
- If you develop another, let me know and I'll add it here!

### Demonstration

There is a [small demo app](https://github.com/JustinAiken/my_timeline-demo) available to show how it looks inside a fresh Rails application with a Devise User system.

### Usage:

1. Add the gem to your Gemfile: `gem 'my_timeline'` and `bundle install`
2. Install the config file: `rails g my_timeline:install`
3. Edit `config/initializers/my_timeline.rb` to taste
4. Mount the engine in your routes:
   ```ruby
   # A timeline belongs_to User
   resources :users do
   mount MyTimeline::Engine => '/timeline', as: :my_timeline
   end
   ```
   or
   ```ruby
   # No Users, just a dedicated timeline route
   mount MyTimeline::Engine => '/timeline', as: :my_timeline
   ```
5. Add a gem for any service you'd like to add on.

### Credits

Original author: [Justin Aiken](https://github.com/JustinAiken)

### Links

- [Source](https://github.com/JustinAiken/my_timeline)
- [Bug Tracker](https://github.com/JustinAiken/my_timeline/issues)
- [Rubygem](https://rubygems.org/gems/my_timeline)

### Note on Patches/Pull Requests

* Fork the project.
* Make your feature addition or bug fix.
* Add tests for it. This is important so I don't break it in a future version unintentionally.
* Commit, do not mess with rakefile, version, or history.
* If you want to have your own version, that is fine but bump version in a commit by itself so I can ignore when I pull.
* Send me a pull request. Bonus points for topic branches.

## Copyright

Copyright (c) 2013 Justin Aiken Inc. MIT license (see LICENSE for details).
Copyright 2013 Justin Aiken

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
# 0.1.0
* [BUGFIX] Fix the settings to reload in development mode
* [BUGFIX] Fix the user stub
* [FEATURE] Display event’s times with the User’s timezone, and make the time formatting configurable
* [BUGFIX] Make the plugin registry a Set instead of an Array to avoid duplicates

# 0.0.5
* [FEATURE] Added a detail view that expands below the summary post, if that model supports an expandable view (defaults to not.)
* [BUGFIX] Dehardcode header text

# 0.0.4
* [FEATURE] Added Rails 4 compatibility
* [FEATURE] Added more tests

# 0.0.3
* [BUGFIX] Fixed external event link

# 0.0.2
* Many, many bugfixes

# 0.0.1
* Initial release
/my_timeline/my_timeline.gemspec

```ruby
$:push File.expand_path("../lib", __FILE__)
require "my_timeline/version"

Gem::Specification.new do |
s
  s.name = "my_timeline"
  s.version = MyTimeline::VERSION
  s.authors = ['Justin Aiken']
  s.email = ['60tonangel@gmail.com']
  s.homepage = "https://www.github.com/JustinAiken/my_timeline"
  s.summary = "Social Media Aggregation Timeline"
  s.description = "Social Media Aggregation Timeline"
  s.license = 'MIT'
  s.files = `git ls-files`.split("\n")
  s.test_files = `git ls-files -- {spec}/#`.split("\n")
  s.add_runtime_dependency "rails", '>= 2.0'
  s.add_runtime_dependency "kaminari"
  s.add_runtime_dependency 'ledermann-rails-settings'
  s.add_development_dependency 'sqlite3'
  s.add_development_dependency 'rspec-rails'
  s.add_development_dependency 'capybara'
  s.add_development_dependency 'factory_girl_rails'
  s.add_development_dependency 'guard-rspec'
end
```
/my_timeline/Gemfile

  1  source "https://rubygems.org"
  2  gemspec
  3  # jquery-rails is used by the dummy application
  4  gem "jquery-rails"
#!/usr/bin/env rake
begin
  require 'bundler/setup'
rescue LoadError
  puts 'You must `gem install bundler` and `bundle install` to run rake tasks'
end

begin
  require 'rdoc/task'
rescue LoadError
  require 'rake/rdoctask'
end
RDoc::Task = Rake::RDocTask

RDoc::Task.new(rdoc) do |rdoc|
  rdoc.rdoc_dir = 'rdoc'
  rdoc.title = 'MyTimeline'
  rdoc.options << '-line-numbers'
  rdoc.rdoc_files.include('README.rdoc')
  rdoc.rdoc_files.include('lib/**/*.rb')
end

APP_RAKEFILE = File.expand_path("../spec/dummy/Rakefile", __FILE__)
load 'rails/tasks/engine.rake'
Bundler::GemHelper.install_tasks

Dir[File.join(File.dirname(__FILE__), 'tasks/**/*.rake')].each { |f| load f }
require 'rspec/core'
require 'rspec/core/rake_task'

desc "Run all specs in spec directory (excluding plugin specs)"
RSpec::Core::RakeTask.new spec: 'app:db:test:prepare'
task default: :spec
/my_timeline/Guardfile

1 guard :rspec cmd: 'zeug rspec --color --format nested --fail-fast', all_after_pass: false, all_on_start: false do
2   watch(%r{^spec/.+_spec.rb$})
3   watch(%r{^lib/(.+).rb$}) { m "spec/lib/#{m[1]}_spec.rb" }
4   watch('spec/spec_helper.rb') { "spec" }
5
6   # Rails
7   watch(%r{^app/(.+).rb$}) { m "spec/#{m[1]}_spec.rb" }
8   watch(%r{^app/(.+)(.erb|haml|slim)$}) { m "spec/#{m[1]}{m[2]}_spec.rb" }
9   watch(%r{^app/controllers/(.+)(controller).rb$}) { m "spec/routing/#{m[1]}_routing_spec.rb" }
10  watch(%r{^spec/support/(.+).rb$}) { "spec" }
11  watch('config/routes.rb') { "spec/routing" }
12  watch('app/controllers/application_controller.rb') { "spec/controllers" }
13
14   # Capybara features
15   watch(%r{^app/views/(.+).*(erb)$}) { m "spec/features/#{m[1]}_spec.rb" }
16 end
"command": "ruby -rubygems -r/engine_plan -eZeus.go",

"plan": {
  "boot": {
    "default_bundle": {
      "development_environment": {
        "prerake": {
          "rake": []
        },
        "runner": ["r"],
        "console": ["c"],
        "generate": ["g"],
        "destroy": ["d"]
      },
      "test_environment": {
        "test_helper": {
          "test": ["rspec"]
        }
      }
    }
  }
}

/my_timeline/.gitignore

1 .bundle/
2 log/*.log
3 pkg/
4 spec/dummy/db/*.sqlite3
5 spec/dummy/log/*.log
6 spec/dummy/tmp/
7 spec/dummy/.sass-cache
8 Gemfile.lock
9 .ruby-version
10 .ruby-gemset
11
language: ruby
rvm:
  - 1.9.3
  - 2.0.0
env:
  - DB=sqlite
gemfile:
  - gemfiles/Gemfile.rails-3.x
  - gemfiles/Gemfile.rails-4.x
script:
  - RAILS_ENV=test bundle exec rake db:migrate
  - bundle exec rake
before_script:
require 'zeus/rails'

ROOT_PATH = File.expand_path(Dir.pwd)
ENV_PATH = File.expand_path('spec/dummy/config/environment', ROOT_PATH)
BOOT_PATH = File.expand_path('spec/dummy/config/boot', ROOT_PATH)
APP_PATH = File.expand_path('spec/dummy/config/application', ROOT_PATH)
ENGINE_ROOT = File.expand_path(Dir.pwd)
ENGINE_PATH = File.expand_path('lib/my_timeline/engine', ENGINE_ROOT)

class EnginePlan < Zeus::Rails
  end

Zeus.plan = EnginePlan.new
class CreateMyTimelinePosts < ActiveRecord::Migration
  def change
    create_table :my_timeline_posts do |t|
      t.text :full_text
      t.datetime :happened_on
      t.references :event
      t.timestamps
    end
  end
end
/my_timeline/db/migrate/20131027171920_create_my_timeline_events.rb

class CreateMyTimelineEvents < ActiveRecord::Migration
  def change
    create_table :my_timeline_events do |t|
      t.text :description
      t.datetime :happened_on
      t.string :icon_name
      t.string :external_link
      t.string :original_id
      t.boolean :public, default: true
      t.integer :importance, default: 5
      t.references :user
      t.references :linkable, polymorphic: true
      t.timestamps
    end
  end
end
/my_timeline/db/migrate/20131103000200_create_my_timeline_settings.rb

class CreateMyTimelineSettings < ActiveRecord::Migration

  def change
    create_table :my_timeline_settings do |t|
      t.string :var, :null => false
      t.text :value
      t.references :target, :null => false, :polymorphic => true
      t.timestamps
    end

    add_index :my_timeline_settings, [:target_type, :target_id, :var], :unique => true, :name => ""
  end

end
<h3>New Post</h3>
<% render partial: "form" %>
<%= form_for @post do |f| %>

<% if @post.errors.any? %>
  <div class="alert alert-error">
    <h2><%= pluralize(@post.errors.count, "error") %> prohibited this post from being saved:</h2>
    <ul>
    <% @post.errors.full_messages.each do |msg| %>
      <li><%= msg %></li>
    <% end %>
    </ul>
  </div>
<% end %>

<% f.label :happened_on, 'Happened On' %>
<% f.datetime_select :happened_on, {prompt: { day: 'Select day', month: 'Select month', year: nil }} %>

<% f.fields_for :event do |ff| %>
  <div>
    <% ff.label :description, 'Description' %>
    <% ff.text_field :description %>
    <div>
      <% ff.label :public, 'Publically viewable' %>
      <% ff.check_box :public %>
    </div>
  </div>
<% end %>

<% f.label :full_text, 'Text:' %>
<% f.text_area :full_text %>

<% f.submit %>
<% end %>
<%= form_for @event do |f| %>

<% if @event.errors.any? %>
  <div class="alert alert-error">
    <h2><%= pluralize(@event.errors.count, "error") %> prohibited this event from being saved:</h2>
    <ul>
      <% @event.errors.full_messages.each do |msg| %>
        <li><%= msg %></li>
      <% end %>
    </ul>
  </div>
<% end %>

<%= f.label :happened_on, 'Happened On' %>
<%= f.text_field :happened_on %>
<br />

<%= f.label :description, 'Description' %>
<%= f.text_field :description %>
<br />

<%= f.label :public, 'Publically viewable' %>
<%= f.check_box :public %>
<br />

<%= f.submit %>
<% end %>
<% javascript_include_tag 'my_timeline/events' %>

<% if t "my_timeline.timeline_header", default: nil %>
  <div class="page-header">
    <small><%= t "my_timeline.timeline_header" %></small>
  </div>
<% end %>

<% @dates_with_events.each do |day| %>
  <h4><%= date_header_string[day[:date]] %></h4>
  <%= render partial: "my_timeline/events/day_with_events_#{MyTimeline.render_method}", locals: {events: day[:events]} %>
<% end %>

<% paginate @events %>

---

/my_timeline/app/views/my_timeline/events/index.html.erb

---

1. `<% javascript_include_tag "my_timeline/events" %>`
2. `<% if t "my_timeline.timeline_header", default: nil %>`
3. `<div class="page-header">
4.   <small><%= t "my_timeline.timeline_header" %></small>
5. </div>
6. `<% end %>`
7. `<% @dates_with_events.each do |day| %>`
8. `<h4><%= date_header_string[day[:date]] %></h4>`
9. `<%= render partial: "my_timeline/events/day_with_events_#{MyTimeline.render_method}", locals: {events: day[:events]} %>`
10. `<% end %>`
11. `<% paginate @events %>

---

---

---
<p> 1</p>
<pre><code>&lt;% if event.linkable.class.respond_to?(:is_exandable) %&gt;
 2  
 3 &lt;%&gt; link_to "#", class: "event_expand", id: "event_#{event.id}" do %&gt;
 4  
 5 &lt;% end %&gt;
 6  
 7 &lt;%&gt; link_to event.external_link do %&gt;
 8 &lt;%&gt; image_tag event.icon_path. size: "32x32" %&gt;
 9 &lt;% end %&gt;
 10 &lt;%&gt; event.happened_on %&gt;
 11 &lt;%&gt; raw event.description %&gt;
 12 &lt;% if @owner_viewing %&gt;
 13 &lt;%&gt; link_to edit_event_path(event) do %&gt;
 14 &lt;%&gt; glyph 'pencil' %&gt;
 15 &lt;% end %&gt;
 16 &lt;%&gt; link_to event_path(event.event), method: :delete do %&gt;
 17 &lt;%&gt; glyph 'remove-sign' %&gt;
 18 &lt;% end %&gt;
 19 &lt;% end %&gt;
 20 &lt;/p&gt;
 21 &lt;% if event.linkable.class.respond_to?(:is_exandable) %&gt;
 22 &lt;div class="event_details" id="event_<%= event.id %>"> style="display:none;"&gt;
 23 &lt;/div&gt;
 24 &lt;span class="event_url" id="event_url_<%= event.id %>"> style="display:none;"&gt;
 25 &lt;%&gt; polymorphic_url [my_timeline, event.linkable] %&gt;
 26 &lt;/span&gt;
 27 &lt;% end %&gt;
 28</code></pre>
/my_timeline/app/views/my_timeline/events/_day_with_events_list.html.erb

1  <ul>
2    <%- events.each do |event| %>
3      <li><%= render partial: 'my_timeline/events/event', object: MyTimeline::EventPresenter.new(event) %></li>
4    <%- end %>
5  </ul>
6
7
<table class="<%= MyTimeline.table_class %>">
  <thead>
    <th></th>
  </thead>
  <tbody>
    <% events.each do |event| %>
      <tr>
        <td style="vertical-align:middle;">
          <% render partial: 'my_timeline/events/event', object: MyTimeline::EventPresenter.new(event) %>
        </td>
      </tr>
    <% end %>
  </tbody>
</table>
<div class="page-header">
  <h1><%= I18n.t("my_timeline.control_panel.header") %></h1>
</div>

<h3>My Settings</h3>
<%= render partial: "my_timeline/control_panel/time_zone" %>

<hr>

<% @enabled_plugins.each do |plug| %>
  <%= render partial: "my_timeline/#{plug}/control_panel" %>
<% end %>

<hr>

<h3>New Post</h3>
<%= link_to "Add", new_post_path %>

a self-contained post.
<%= form_for @user, url: control_panel_timezone_path, method: :post do %>
  <%= f.time_zone_select :time_zone %>
  <br>
  <%= f.submit 'Save', class: "btn btn-primary" %>
<% end %>
$("a.event_expand").click (event) ->
event.preventDefault()
event_id = $(this).attr("id").replace /[A-Z_a-z$-]/g, ""
div_id = "div#event_" + event_id
url = $("span#event_url_" + event_id).text()
if $(div_id).css("display") == 'none'
  $.ajax url
    type: 'GET'
    dataType: 'html'
    error: (jqXHR, textStatus, errorThrown) ->
      $(div_id).html errorThrown
      $(div_id).toggle()
    success: (data, textStatus, jqXHR) ->
      $(div_id).html data
      $(div_id).toggle()
else
  $(div_id).toggle()
/*
 * This is a manifest file that'll be compiled into application.css, which will include all the files
 * listed below.
 *
 * Any CSS and SCSS file within this directory, lib/assets/stylesheets, vendor/assets/stylesheets,
 * or vendor/assets/stylesheets of plugins, if any, can be referenced here using a relative path.
 *
 * You're free to add application-wide styles to this file and they'll appear at the top of the
 * compiled file, but it's generally better to create a new file per style scope.
 *
 /*= require_self
 /*= require_tree.
 */
module MyTimeline
  class Post < ActiveRecord::Base
    unless rails4?
      attr_accessible :happened_on, :full_text
      attr_accessible :event, :event_id, :event_attributes
    end
    belongs_to :event, dependent: :destroy
    validates :happened_on, presence: true
    validates :full_text, presence: true
    accepts_nested_attributes_for :event
    def self.is_exandable?
      true
    end
  end
end
module MyTimeline

class Event < ActiveRecord::Base

  unless rails4?
    attr_accessible :description, :happened_on, :icon_name, :external_link, :original_id, :public, :importance
  else
    attr_accessible :user, :linkable, :user_id, :linkable_type, :linkable_id
  end

  belongs_to :linkable, polymorphic: true, dependent: :delete
  belongs_to :user, class_name: MyTimeline.user_class.to_s

  validates :description, presence: true
  validates :happened_on, presence: true
  validates :importance, inclusion: { in: 1..10, allow_blank: true, message: "%{value} is not between 1-10." }

  scope :desc, order("my_timeline_events.happened_on DESC")

end
end
module MyTimeline
  module EventsHelper
    def date_header_string(date)
      date.strftime "%b #{date.day.ordinalize}, %Y"
    end
  end
end
module MyTimeline
  module ApplicationHelper
    def method_missing(meth, *args, &block)
      if meth.to_s =~ /_path$|_url$/
        if main_app.respond_to? meth
          main_app.send meth, *args
        else
          super
        end
      else
        super
      end
    end
    def respond_to?(meth)
      if meth.to_s =~ /_path$|_url$/
        if main_app.respond_to? meth
          true
        else
          super
        end
      else
        super
      end
    end
  end
end
module MyTimeline
  class EventPresenter
    attr_accessor :event

    def initialize(event)
      @event = event
    end

    def icon_path
      "my_timeline/icons/#{event.icon_name}"
    end

    def happened_on
      time.strftime MyTimeline.time_formatter
    end

    def id
      event.id
    end

    private

    def time
      return event.happened_on if event.linkable.class.respond_to? :keep_original_time_zone?
      return event.happened_on unless user.time_zone.present?
      event.happened_on.in_time_zone user.time_zone
    end

    def user
      if MyTimeline.user_class == MyTimeline::UserStub
        MyTimeline::UserStub
      else
        event.user
      end
    end

    def method_missing(meth, *args, &blk)
      if event.respond_to?(meth)
        event.send meth, *args
      else
        super
      end
    end
  end
end
module MyTimeline
  class PostsController < MyTimeline::ApplicationController
    def new
      @event = Event.new
      @post = Post.new(event: @event)
    end

    def create
      @post = Post.new(rails4? ? post_params : params[:post])
      @post.event.happened_on = @post.happened_on
      @post.event.user_id     = @user.id
      @post.event.icon_name   = "notes.png"
      if @post.save
        @post.event.linkable = @post
        @post.event.save
        redirect_to root_path, notice: "Post saved."
      else
        render :new
      end
    end

    def show
      @post = Post.find_by_id params[:id]
      render text: @post.full_text
    end

    private

    if rails4?
      define_method :post_params do
      end
    end
  end
end
module MyTimeline

class EventsController < MyTimeline::ApplicationController

def index
  if @show_hidden
    q = {}
  else
    q = { public: true }
  end

  @events = @user.events.where(q).desc.page params[:page]

  @events_by_day = @events.all.to_a.group_by { |e| e.happened_on.to_date }
  @dates_with_events = build_dates
end

def show
  #
end

def edit
  @event = Event.find_by_id params[:id]
end

def update
  @event = Event.find_by_id params[:id]
  if @event.update_attributes(rails4? ? event_params : params[:event])
    redirect_to root_path, notice: "Edit successful."
  else
    render 'edit'
  end
end

def destroy
  @event = Event.find_by_id(params[:id])
  @event.destroy
  redirect_to root_path
end

private

DateWithEvents = Struct.new(:date, :events)
def build_dates
  [].tap do |array|
    @events_by_day.each do |date, events|
      array << DateWithEvents.new(date, events.reverse)
    end
  end
end

if rails4?
  define_method :event_params do
    params.required(:event).permit :description, :happened_on, :public
  end
end
class MyTimeline::ApplicationController < ApplicationController
  before_filter :find_user
  private
  def find_user
    @user = MyTimeline.user_class.send "find_by_#{MyTimeline.user_slug}".params[:user_id]
    if @user == current_user
      @owner_viewing = true
      @show_hidden = true
    else
      #
    end
    params.delete :user_id
  end
end
module MyTimeline
  class ControlPanelController < MyTimeline::ApplicationController

    before_filter :user_only

    def index
      @enabled_plugins = MyTimeline.enabled_plugins
    end

def timezone
  if rails4?
    @user.time_zone = user_params[:time_zone]
  else
    @user.time_zone = params[user_param][:time_zone]
  end

  @user.save!
  redirect_to :back, notice: "Time zone setting saved."
end

private

if rails4?
def user_params
  define_method :user_params do
    params.required(user_param).permit :time_zone
  end
end
end

def user_param
  MyTimeline.user_class.model_name.param_key.to_sym
end

def user_only
  unless @owner_viewing
    redirect_to root_path, notice: "Can’t see that!"
  end
end
end
end
require 'kaminari'
require '"my_timeline/user_stub"'
require '"my_timeline/engine"

module MyTimeline

  mattr_accessor :user_class, :user_slug, :render_method, :table_class, :config_object, :enabled_plugins, :time_formatter

  @@user_class = 'MyTimeline::UserStub'
  def self.user_class
    @@user_class.constantize
  end

  @@user_slug = :id
  @@render_method = 'table'
  @@use_bootstrap = true

  @@table_class = "table table-striped"
  @@time_formatter = "%-l:%M %P - "

  @@enabled_plugins = Set.new
  def self.register_plugin(plugin_name, options = {}) MyTimeline.config_object[:plugin_name].options
    @@enabled_plugins << plugin_name
  end

  def self.setup
    yield self
  end
end
<file>

/my_timeline/lib/tasks/my_timeline_tasks.rake

1  # desc "Explaining what the task does"
2  # task :my_timeline do
3  #   # Task goes here
4  # end
</file>
Please add my_timeline to your routes file:

1. For a multiple-user instance:

resources :users do
  mount MyTimeline::Engine => '/timeline', as: :my_timeline
end

2. For a single timeline:

mount MyTimeline::Engine => '/timeline', as: :my_timeline
/my_timeline/lib/generators/templates/my_timeline.rb

MyTimeline.setup do |config|
  # The User class to use... Default is "User".
  # Set to nil to not use per-user timelines,
  # or put a constant in a string to use that class
  config.user_class = "User"

  # By default, looks for the user by id, but if you want to use a name or a slug,
  # set it here. I.E., config.user_slug = :nick_name would result in User.find_by_nick_name
  #config.user_slug = :id

  # How to render the events - in a :table, or in a :list
  # config.render_method = :table

  # What classes to style the table with
  # config.table_class = "table table-striped"

  # How to format the time of the event; default looks like "3:33 pm - "
  # config.time_formatter = %l:%M %P - "
end
module MyTimeline
  module Generators
    class InstallGenerator < Rails::Generators::Base
      source_root File.expand_path("../../templates", __FILE__)
      desc "Creates a MyTimeline initializer and copy locale files to your application."
      class_option :orm
      def copy_initializer
        template "my_timeline.rb", "config/initializers/my_timeline.rb"
      end
      def show_readme
        readme "README" if behavior == :invoke
      end
      end
    end
  end
end
/my_timeline/lib/my_timeline/engine.rb

require 'my_timeline/settings_ext'
require 'my_timeline/core_ext/rails4'

module MyTimeline
  class Engine < ::Rails::Engine
    isolate_namespace MyTimeline

    config.autoload_paths << File.expand_path("./app/classes/**", __FILE__)
    config.autoload_paths << File.expand_path("./app/scrapers/**", __FILE__)

    config.generators do |g|
      g.test_framework :rspec, fixture: false
      g.fixture_replacement :factory_gir, dir: 'spec/factories'
      g.assets false
      g.helper false
    end

    config.after_initialize do |app|
      MyTimeline::SettingsExt.extend_rails_settings
    end

    config.to_prepare do |app|
      MyTimeline::SettingsExt.extend_rails_settings
    end if Rails.env.development?
  end
end
module MyTimeline
  VERSION = "0.1.0"
end
module MyTimeline

class UserStub
    include Singleton

    include ActiveModel::Validations
    include ActiveModel::Conversion
    extend ActiveModel::Naming

    def events
        Event
    end

    def settings(var = :core)
        RailsSettings::SettingObject.find_by_var var
    end

    def id
        nil
    end

    def save!
        true
    end

    def persisted?
        false
    end

    def self.method_missing(meth, *args, &blk)
        if meth.to_s =~ /^find_by/ UserStub else
            instance.send meth, *args, &blk
            # super
        end
    end

    def self.settings_attr_accessor(*args)
        args.each do |method_name|
            eval "
                def self.#{method_name.to_s}
                    RailsSettings::SettingObject.find_by_var(:core).send("#{method_name}")
                end
                def self.#{method_name.to_s}=(value)
                    RailsSettings::SettingObject.find_by_var(:core).send("#{method_name}='", value)
                end
            "
        end
    end

    settings_attr_accessor :time_zone
end
end
require 'rails-settings'

module MyTimeline
  module SettingsExt
    def self.extend_rails_settings
      RailsSettings::SettingObject.class_eval do
        self.table_name = "my_timeline_settings"
        self.config_object = ::RailsSettings::Configuration.new(MyTimeline::User.new) do
          s.key :core
        end
        MyTimeline::User.class_eval do
          self.send :include, ::RailsSettings::Base
          self.send :extend, ::RailsSettings::Scopes
          def self.settings_attr_accessor(*args)
            args.each do |method_name|
              eval "def #{method_name}
                self.settings(:core).send(:#{method_name})
              end
              def #{method_name}=(value)
                self.settings(:core).send(:#{method_name}=, value)
              end
            end
            settings_attr_accessor :time_zone
          end unless MyTimeline::User == MyTimeline::UserStub
        end
      end
    end
  end
end
module Kernel
  define_method :rails4? do
    Rails::VERSION::MAJOR >= 4
  end
end
ENV['RAILS_ENV'] ||= 'test'
require File.expand_path '../dummy/config/environment.rb', __FILE__
require 'rspec/rails'
require 'factory_girl_rails'
Rails.backtrace_cleaner.remove_silencers!

# Load support files
Dir['#{File.dirname(__FILE__)}/support/**/*.rb'].each { |f| require f }
RSpec.configure do |config|
  config.include MyTimeline::Engine.routes.url_helpers
  config.before(:each) { @routes = MyTimeline::Engine.routes }
  config.mock_with :rspec
  config.use_transactional_fixtures = true
  config.infer_base_class_for_anonymous_controllers = false
  config.order = "random"
  config.filter_run focus: true
  config.run_all_when_everything_filtered = true
end
# Add your own tasks in files placed in lib/tasks ending in .rake,
# for example lib/tasks/capistrano.rake, and they will automatically be available to Rake.

require File.expand_path('..config/application', __FILE__)

Dummy::Application.load_tasks
# This file is used by Rack-based servers to start the application.

```ruby
require ::File.expand_path('..config/environment', __FILE__)
run Dummy::Application
```
# This file is auto-generated from the current state of the database. Instead
# of editing this file, please use the migrations feature of Active Record to
# incrementally modify your database, and then regenerate this schema definition.

# Note that this schema.rb definition is the authoritative source for your
database schema. If you need to create the application database on another
system, you should be using db:schema:load, not running all the migrations
from scratch. The latter is a flawed and unsustainable approach (the more migrations
you'll amass, the slower it'll run and the greater likelihood for issues).

# It's strongly recommended to check this file into your version control system.

ActiveRecord::Schema.define(version => 20131103135539) do
  create_table "my_timeline_events", :force => true do |
    t.string "description"
    t.datetime "happened_on"
    t.string "icon_name"
    t.string "external_link"
    t.string "original_id"
    t.boolean "public", :default => true
    t.integer "importance", :default => 5
    t.integer "user_id"
    t.integer "linkable_id"
    t.string "linkable_type"
    t.datetime "created_at", :null => false
    t.datetime "updated_at", :null => false
  end
  create_table "my_timeline_foos", :force => true do |
    t.string "name"
    t.datetime "created_at", :null => false
    t.datetime "updated_at", :null => false
  end
  create_table "my_timeline_posts", :force => true do |
    t.text "full_text"
    t.datetime "happened_on"
    t.integer "event_id"
    t.datetime "created_at", :null => false
    t.datetime "updated_at", :null => false
  end
  create_table "my_timeline_settings", :force => true do |
    t.string "var", :null => false
    t.text "value"
    t.integer "target_id", :null => false
    t.string "target_type", :null => false
    t.datetime "created_at", :null => false
    t.datetime "updated_at", :null => false
  end
  add_index "my_timeline_settings", ["target_type", "target_id", "var"], :name => "index_my_timeline_settings_on_user", :unique => true
end
require 'rubygems'
gemfile = File.expand_path('../../../../Gemfile', __FILE__)
if File.exist?(gemfile)
  ENV['BUNDLE_GEMFILE'] = gemfile
  require 'bundler'
  Bundler.setup
end
$:unshift File.expand_path('../../../../lib', __FILE__)
Rails.application.routes.draw do
  mount MyTimeline::Engine => '/my_timeline'
end
/my_timeline/spec/dummy/config/database.yml

1   test:
2     adapter: sqlite3
3     database: db/test.sqlite3
4     pool: 5
5     timeout: 5000
6   development:
7     adapter: sqlite3
8     database: db/development.sqlite3
9     pool: 5
10    timeout: 5000
11
/my_timeline/spec/dummy/config/application.rb

```ruby
require File.expand_path('../boot', __FILE__)
require 'active_record/railtie'
require 'action_controller/railtie'
require 'action_mailer/railtie'
require 'sprockets/railtie'

begin
  require 'active_resource/railtie'
rescue LoadError
end

Bundler.require("Rails::groups")
require "my_timeline"

module Dummy
  class Application < Rails::Application
    config.encoding = "utf-8"
    config.filter_parameters += [:password]
    config.active_support.escape_html_entities_in_json = true
    config.active_record.whitelist_attributes = true unless rails4?
    config.assets.enabled = true
    config.assets.version = '1.0'
  end
end
```

# Load the rails application
require File.expand_path('..:/application', __FILE__)

# Initialize the rails application
Dummy::Application.initialize!
Dummy::Application.configure do
  # Settings specified here will take precedence over those in config/application.rb
  # The test environment is used exclusively to run your application's
  # test suite. You never need to work with it otherwise. Remember that
  # your test database is "scratch space" for the test suite and is wiped
  # and recreated between test runs. Don't rely on the data there!
  config.cache_classes = true

  # Configure static asset server for tests with Cache-Control for performance
  config.serve_static_assets = true
  config.static_cache_control = "public, max-age=3600"

  # Log error messages when you accidentally call methods on nil
  config.whiny_nils = true unless rails4?

  # Show full error reports and disable caching
  config.consider_all_requests_local = true
  config.action_controller.perform_caching = false

  # Raise exceptions instead of rendering exception templates
  config.action_dispatch.show_exceptions = false

  # Disable request forgery protection in test environment
  config.action_controller.allow_forgery_protection = false

  # Tell Action Mailer not to deliver emails to the real world.
  # The :test delivery method accumulates sent emails in the
  # ActionMailer::Base.deliveries array.
  config.action_mailer.delivery_method = :test

  # Raise exception on mass assignment protection for Active Record models
  config.active_record.mass_assignment_sanitizer = :strict unless rails4?

  # Print deprecation notices to the stderr
  config.active_support.deprecation = :stderr
end
/my_timeline/spec/dummy/config/environments/development.rb

```ruby
Dummy::Application.configure do
  # Settings specified here will take precedence over those in config/application.rb

  # In the development environment your application's code is reloaded on
  # every request. This slows down response time but is perfect for development
  # since you don't have to restart the web server when you make code changes.
  config.cache_classes = false

  # Log error messages when you accidentally call methods on nil.
  config.whiny_nils = true unless rails4?

  # Show full error reports and disable caching
  config.consider_all_requests_local = true
  config.action_controller.perform_caching = false

  # Don't care if the mailer can't send
  config.action_mailer.raise_delivery_errors = false

  # Print deprecation notices to the Rails logger
  config.active_support.deprecation = :log

  # Only use best-standards-support built into browsers
  config.action_dispatch.best_standards_support = :builtin

  # Raise exception on mass assignment protection for Active Record models
  config.active_record.mass_assignment_sanitizer = :strict unless rails4?

  # Log the query plan for queries taking more than this (works
  # with SQLite, MySQL, and PostgreSQL)
  config.active_record.auto_explain_threshold_in_seconds = 0.5 unless rails4?

  # Do not compress assets
  config.assets.compress = false

  # Expands the lines which load the assets
  config.assets.debug = true
end
```
#!/usr/bin/env ruby
#
# This command will automatically be run when you run "rails" with Rails 3 gems installed from the root of your application.

APP_PATH = File.expand_path('../../config/application', __FILE__)
require File.expand_path('../../config/boot', __FILE__)
require 'rails/commands'
require 'spec_helper'

module MyTimeline
  describe Event do
    it "is an event" do
      e = Event.new
      e.save.should be_false

      f = Event.new(
        happened_on: Time.now,
        description: "foo",
        original_id: "6"
      )
      f.save.should be_true

      e = FactoryGirl.create(:my_timeline_event)
      e.save.should be_true
    end
  end
end
module MyTimeline
  describe ApplicationHelper do
    class HelperTester
      include MyTimeline::ApplicationHelper
    end
    class MainApp
      def self.bar_path
        '/bar'
      end
    end
def main_app
      MainApp
    end
  end
subject { HelperTester.new }

describe "#method_missing" do
  it "with a non url/path method, herps and dies" do
    expect { subject.foo_and_bar }.to raise_error NoMethodError
  end
  it "with a url/path method not found in the main app, herps and dies" do
    expect { subject.foo_path }.to raise_error NoMethodError
  end
  it "passes a valid url/path method back to the main app" do
    expect(subject.bar_path).to eq '/bar'
  end
end
FactoryGirl.define do
  factory :my_timeline_event, :class => MyTimeline::Event do
    happened_on { Time.now - 1.year }
    description "Foo"
    original_id "1"
  end
end
require 'spec_helper'

describe MyTimeline::PostsController do
  routes { MyTimeline::Engine.routes }
  before { ApplicationController.any_instance.stub :current_user }

  describe 'GET #new' do
    it "news" do
      get "new"
      subject.instance_variable_get(:@post).should be_a MyTimeline::Post
      subject.instance_variable_get(:@event).should be_a MyTimeline::Event
    end
  end

  describe 'POST #create' do
    it "creates the event/post" do
      post "create", post: { happened_on: Time.now, full_text: "foo", event_attributes: { description: "bar", public: "true" } }
      MyTimeline::Post.last.should_not be_nil
      response.should redirect_to root_path
    end
  end
end
describe MyTimeline::EventsController do
  routes { MyTimeline::Engine.routes }
  before { ApplicationController.any_instance.stub :current_user }
  describe "GET #index" do
    it "gets okay" do
      get "index"
      response.code.should == "200"
    end
  end
  describe "GET #show" do
    it "shows stuff" do
    end
  end
  describe "GET #edit" do
    it "edits" do
      get "edit"
    end
  end
  describe "#update" do
    it "updates" do
    end
  end
  describe "#destroy" do
    it "kills it" do
    end
  end
end
require 'spec_helper'

describe MyTimeline:: ApplicationController do
  describe '#find_user' do
    before { subject.params = {user_id: '7'} }
    after do
      subject.params.should == []
    end
    context "when the user exists" do
      it "sets some variables" do
        MyTimeline::UserStub.should_receive(:find_by_id).with('7').and_return('foo'
        subject.stub(:current_user).and_return 'foo'
        subject.send :find_user
        subject.instance_variable_get(:@owner_viewing).should be_true
        subject.instance_variable_get(:@show_hidden).should be_true
      end
    end
    context "when the user is not found" do
      it "doesn't do much" do
        subject.stub(:current_user).and_return nil
        subject.send :find_user
        subject.instance_variable_get(:@owner_viewing).should be_false
        subject.instance_variable_get(:@show_hidden).should be_false
      end
    end
  end
end
require 'spec_helper'

describe MyTimeline::ControlPanelController do
  routes { MyTimeline::Engine.routes }

  before { ApplicationController.any_instance.stub :current_user }

  describe 'GET #index' do
    it "gets okay" do
      get "index"
      response.code.should == "200"
    end
  end
end
/my_timeline/config/routes.rb

```
MyTimeline::Engine.routes.draw do
  root to: "events#index"
  resources :events
  resources :posts
  get "control_panel" => "control_panel#index", as: "control_panel"
  post "control_panel" => "control_panel#timezone", as: "control_panel_timezone"
end
```
/my_timeline/config/locales/en.yml

1  en:
2    my_timeline:
3      timeline_header: A demonstration timeline
4    control_panel:
5      header: Control Panel
6
/my_timeline/script/rails

#!/usr/bin/env ruby
#
# This command will automatically be run when you run "rails" with Rails 3 gems installed from the root of your application.

ENGINE_ROOT = File.expand_path('..', __FILE__)
ENGINE_PATH = File.expand_path('../../lib/my_timeline/engine', __FILE__)

require 'rails/all'
require 'rails/engine/commands'

require 'rails/all'
require 'rails/engine/commands'
/my_timeline/gemfiles/Gemfile.rails-3.x

1 source "https://rubygems.org"
2 gemspec :path => '..'
3 gem "jquery-rails"
4 gem "rails", '~>3.2"
source "https://rubygems.org"
gemspec :path => '..'
gem "jquery-rails"
gem "rails", ">4.0"
# My Timeline-Demo

### A social-media aggregation/display plugin display application

This is a small demo app to host the [My Timeline](https://github.com/JustinAiken/my_timeline) Rails application.

It lets you quickly see what it looks like, including when it's styled with a myriad of [free bootstrap themes](http://bootswatch.com/).

### What it looks like:
![Screenshot](doc/screenshot.png)

### Usage:

1. Clone the project (`git clone git@github.com:JustinAiken/my_timeline-demo.git`)
2. Bundle the gems (`cd my_timeline-demo && bundle install`)
3. Prepare the database (`db:create && db:migrate && db:seed`)
4. Put in your own keys in `config/initializers/my_timeline.rb` (config/initializers/my_timeline.rb)
5. Start the rails server (`rails s`)
6. Visit [the website](http://127.0.0.1:3000) in a browser
7. Login with the user created during the seed step
8. ???
9. Profit!

## Credits

Original author: [Justin Aiken](https://github.com/JustinAiken)

## Links

* [Source](https://github.com/JustinAiken/my_timeline-demo)
* [Bug Tracker](https://github.com/JustinAiken/my_timeline-demo/issues)

## Note on Patches/Pull Requests

* Fork the project.
* Make your feature addition or bug fix.
* Add tests for it. This is important so I don't break it in a future version unintentionally.
* Commit, do not mess with rakefile, version, or history.
* If you want to have your own version, that is fine but bump version in a commit by itself so I can ignore when I pull
* Send me a pull request. Bonus points for topic branches.

## Copyright

Copyright (c) 2013 Justin Aiken MIT license (see LICENSE for details).
Copyright 2013 Justin Aiken

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
"Software"), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE
LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION
WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
/my_timeline-demo/Gemfile

source 'https://rubygems.org'

# Rails Base

gem 'rails', '3.2.15'
gem 'mysql2'
gem 'devise'
gem 'thin'

# My Timeline and My Timeline addons

gem 'my_timeline', path: '/Users/jaiken/projects/timeline/my_timeline'
gem 'my_timeline-health_graph', path: '/Users/jaiken/projects/timeline/my_timeline-health_graph'
gem 'my_timeline-twitter', path: '/Users/jaiken/projects/timeline/my_timeline-twitter'
gem 'my_timeline-github', path: '/Users/jaiken/projects/timeline/my_timeline-github'

# Gems needed by My Timeline Addons

gem 'health_graph', git: 'git://github.com/jupp0r/health_graph.git'

group :assets do
  gem 'sass-rails', '~> 3.2.3'
gem 'coffee-rails', '~> 3.2.1'
gem 'uglifier', '>= 1.0.3'
gem 'twitter-bootstrap-rails'
end

group :development do
  gem 'rails-erd'
gem 'quiet_assets'
gem 'better_errors'
gem 'binding_of_caller'
end

gem 'jquery-rails'
#!/usr/bin/env rake

# Add your own tasks in files placed in lib/tasks ending in .rake,
# for example lib/tasks/capistrano.rake, and they will automatically be available to Rake.

require File.expand_path('../config/application', __FILE__)

TimelineApp::Application.load_tasks
# This file is used by Rack-based servers to start the application.

```
require File.expand_path('../config/environment', __FILE__)
run TimelineApp::Application
```
```
/my_timeline-demo/zeus.json

{
  "command": "ruby -rubygems -r/custom_plan -eZeus.go",
  "plan": {
    "boot": {
      "default_bundle": {
        "development_environment": {
          "prerake": {
            "rake": []
          },
          "runner": ["r"],
          "console": ["c"],
          "server": ["s"],
          "generate": ["g"],
          "destroy": ["d"]
        }
      }
    }
  }
}
```
/my_timeline-demo/.gitignore

# See http://help.github.com/ignore-files/ for more about ignoring files.
# If you find yourself ignoring temporary files generated by your text editor
# or operating system, you probably want to add a global ignore instead:
#   git config --global core.excludesfile ~/.gitignore_global

# Ignore bundler config
/.bundle

# Ignore the default SQLite database.
/db/*.sqlite3

# Ignore all logfiles and tempfiles.
/log/*.log
/tmp

.rake_tasks
.ruby-gemset
.ruby-version
.zeus.sock
remote: git://github.com/jupp0r/health_graph.git
revision: aac3be6dc4ec870d0ddf7445408982e6174e3992

health_graph (0.5.7)
faraday (>= 0.7.4)
faraday_middleware (>= 0.7.8)
hashie (>= 1.2)
oauth2 (>= 0.5.2)
webmock (>= 1.7.6)

remote: /Users/jaiken/projects/timeline/my_timeline
specs:
  my_timeline (0.1.0)
  kaminari
  ledermann-rails-settings
  rails (>= 2.0)

remote: /Users/jaiken/projects/timeline/my_timeline-github
specs:
  my_timeline (0.1.0)
  octokit

remote: /Users/jaiken/projects/timeline/my_timeline-health_graph
specs:
  my_timeline-health_graph (0.1.0)
  health_graph
  my_timeline (>= 0.1.0)

remote: /Users/jaiken/projects/timeline/my_timeline-twitter
specs:
  my_timeline-twitter (0.1.0)
  my_timeline (>= 0.1.0)
twitter (>= 5.0, < 6.0)

remote: https://rubygems.org/
specs:
  actionmailer (3.2.15)
  actionpack (= 3.2.15)
  mail (~> 2.5.4)
  actionpack (3.2.15)
  activemodel (= 3.2.15)
  activesupport (= 3.2.15)
  builder (~> 3.0.0)
erubis (~> 2.7.0)
journey (~> 1.0.4)
rack (~> 1.4.5)
rack-cache (~> 1.2)
rack-test (~> 0.6.1)
sprockets (~> 2.2.1)
activemodel (3.2.15)
activesupport (>= 3.2.15)
builder (~> 3.0.0)
activerecord (3.2.15)
activemodel (3.2.15)
activesupport (>= 3.2.15)
arel (~> 3.0.2)
tzinfo (~> 0.3.29)
activeresource (3.2.15)
activemodel (3.2.15)
activesupport (>= 3.2.15)
activemodel (3.2.15)
activesupport (>= 3.2.15)
/my_timeline-demo/Gemfile.lock

68  activesupport (3.2.15)
69  i18n (>= 0.6, >= 0.6.4)
70  multi_json (>= 1.0)
71  addressable (2.3.5)
72  arel (3.0.2)
73  atomic (1.1.14)
74  bcrypt-ruby (3.1.2)
75  better_errors (1.0.1)
76  coderay (>= 1.0.0)
77  erubis (>= 2.6.6)
78  binding_of_caller (0.7.2)
79  debug_inspector (>= 0.0.1)
80  buftok (0.2.0)
81  builder (3.0.4)
82  choice (0.1.6)
83  coderay (1.1.0)
84  coffee-rails (3.2.2)
85  coffee-script (>= 2.2.0)
86  railties (>= 3.2.0)
87  coffee-script (2.2.0)
88  coffee-script-source
89  execjs
90  coffee-script-source (1.6.3)
91  crack (0.4.1)
92  safe_yaml (>= 0.9.0)
93  daemons (1.1.9)
94  debug_inspector (0.0.2)
95  descendants_tracker (0.0.3)
96  devise (3.1.1)
97  bcrypt-ruby (>= 3.0.0)
98  orm_adapter (>= 0.1)
99  railties (>= 3.2.6, < 5)
100  thread_safe (>= 0.1)
101  warden (>= 1.2.3)
102  equalizer (0.0.9)
103  erb (2.7.0)
104  eventmachine (1.0.3)
105  execjs (2.0.2)
106  faraday (0.8.8)
107  multipart-post (>= 1.2.0)
108  faraday_middleware (0.9.0)
109  faraday (>= 0.7.4, < 0.9)
110  hashie (2.0.5)
111  hike (1.2.3)
112  http (0.5.0)
113  http_parser.rb
114  http_parser.rb (0.6.0)
115  httplib (0.2.0)
116  i18n (0.6.5)
117  journey (1.0.4)
118  jquery-rails (3.0.4)
119  railties (>= 3.0.0, < 5.0)
120  thor (>= 0.14, < 2.0)
121  json (1.8.1)
122  jwt (0.1.8)
123  multi_json (>= 1.5)
124  kaminari (0.15.1)
125  actionpack (>= 3.0.0)
126  activesupport (>= 3.0.0)
127  ledermann-rails-settings (2.2.0)
128  activerecord (>= 3.1)
129  mail (2.5.4)
130  mime-types (>= 1.16)
131  treetop (>= 1.4.8)
132  memoizeable (0.4.0)
133  thread_safe (>= 0.1.3)
134  mime-types (1.25)
/my_timeline-demo/Gemfile.lock

202  thor (0.18.1)
203  thread_safe (0.1.3)
204  atomic
205  tilt (1.4.1)
206  treetop (1.4.15)
207  polyglot
208  polyglot (>= 0.3.1)
209  twitter (5.5.1)
210  addressable (-> 2.3)
211  buftok (-> 0.2.0)
212  descendants_tracker (-> 0.0.3)
213  equalizer (-> 0.0.9)
214  faraday (>= 0.8, < 0.10)
215  http (-> 0.5.0)
216  http_parser.rb (-> 0.6.0)
217  json (-> 1.8)
218  memoizable (-> 0.4.0)
219  simple_oauth (-> 0.2.0)
220  twitter-bootstrap-rails (2.2.8)
221  actionpack (>= 3.1)
222  execjs
223  rails (>= 3.1)
224  railties (>= 3.1)
225  tzinfo (0.3.38)
226  uglifier (2.3.0)
227  execjs (>= 0.3.0)
228  json (-> 1.8.0)
229  warden (1.2.3)
230  rack (-> 1.0)
231  webmock (1.15.2)
232  addressable (>= 2.2.7)
233  crack (-> 0.3.2)
234
235  PLATFORMS
236  ruby
237
238  DEPENDENCIES
239  better_errors
240  binding_of_caller
241  coffee-rails (-> 3.2.1)
242  devise
243  health_graph!
244  jquery-rails
245  my_timeline!
246  my_timeline-github!
247  my_timeline-health_graph!
248  my_timeline-twitter!
249  mysql2
250  quiet_assets
251  rails (= 3.2.15)
252  rails-erd
253  sass-rails (-> 3.2.3)
254  thin
255  twitter-bootstrap-rails
256  uglifier (>= 1.0.3)
257
require 'zeus/rails'

class CustomPlan < Zeus::Rails
  # def my_custom_command
  #   # see https://github.com/burke/zeus/blob/master/docs/ruby/modifying.md
  # end

end

Zeus.plan = CustomPlan.new
`/my_timeline-demo/db/seeds.rb`

```ruby
def get_input(display_name, default_val)
    puts "Please enter #{display_name} (or press enter to use #{default_val}):"
    input = STDIN.gets.chomp
    input = default_val if input.blank?
    input
end

puts "Creating Sample User....."

first_name = get_input("first name", "Foo")
last_name = get_input("last name", "Bar")
email = get_input("email", "foo@bar.com")
password = get_input("password", "foobar")

u = User.create(
    email: email,
    password: password,
    password_confirmation: password,
    first_name: first_name,
    last_name: last_name
)

puts "User = #{u.inspect}"

u.save!

puts "Saved!"
```

```
# This file is auto-generated from the current state of the database. Instead
# of editing this file, please use the migrations feature of Active Record to
# incrementally modify your database, and then regenerate this schema definition.

# Note that this schema.rb definition is the authoritative source for your
# database schema. If you need to create the application database on another
# system, you should be using db:schema:load, not running all the migrations
# from scratch. The latter is a flawed and unsustainable approach (the more migrations
# you'll amass, the slower it'll run and the greater likelihood for issues).

# It's strongly recommended to check this file into your version control system.

ActiveRecord::Schema.define(version: '20131222244425') do
  create_table 'my_timeline_events', force: true do |
    |t| do |
    |t| .text  :description
    |t| .datetime  :happened_on
    |t| .string  :icon_name
    |t| .string  :external_link
    |t| .string  :original_id
    |t| .boolean  :public
    |t| .integer  :importance
    |t| .integer  :user_id
    |t| .integer  :linkable_id
    |t| .string  :linkable_type
    |t| .datetime  :created_at
    |t| .datetime  :updated_at
  end

  create_table 'my_timeline_github_fork_events', force: true do |
    |t| do |
    |t| .datetime  :happened_on
    |t| .string  :original_id
    |t| .string  :repo
    |t| .integer  :event_id
    |t| .datetime  :created_at
    |t| .datetime  :updated_at
  end

  create_table 'my_timeline_health_graph_cardio_activities', force: true do |
    |t| do |
    |t| .datetime  :happened_on
    |t| .float  :meters
    |t| .float  :duration
    |t| .integer  :calories
    |t| .string  :routefile
    |t| .string  :uri
    |t| .text  :notes
    |t| .string  :equipment
    |t| .float  :climb
    |t| .string  :activity_type
    |t| .integer  :event_id
    |t| .datetime  :created_at
    |t| .datetime  :updated_at
  end

  create_table 'my_timeline_posts', force: true do |
    |t| do |
    |t| .text  :full_text
    |t| .datetime  :happened_on
    |t| .integer  :event_id
    |t| .datetime  :created_at
    |t| .datetime  :updated_at
  end

  create_table 'my_timeline_settings', force: true do |
    |t| do |
    |t| .string  :var
    |t| .text  :value
    |t| .integer  :target_id
  end
end
/my_timeline-demo/db/schema.rb

```ruby
68  t.string  "target_type",  :null => false
69  t.datetime "created_at",  :null => false
70  t.datetime "updated_at",  :null => false
71  end
72
73  add_index  "my_timeline_settings",  ["target_type",  "target_id",  "var"],  :name => "index_my_timeline_settings_on_target_type_and_target_id_and_var",  :unique => true
74
75  create_table  "my_timeline_twitter_tweets",  :force => true do |t|
76    t.datetime  "happened_on"
77    t.text    "uri"
78    t.text    "post"
79    t.integer  "event_id"
80    t.datetime  "created_at",  :null => false
81    t.datetime  "updated_at",  :null => false
82  end
83
84  create_table  "users",  :force => true do |t|
85    t.datetime  "created_at",  :null => false
86    t.datetime  "updated_at",  :null => false
87    t.string  "email",  :default => "",  :null => false
88    t.string  "encrypted_password",  :default => "",  :null => false
89    t.string  "first_name"
90    t.string  "last_name"
91    t.string  "reset_password_token"
92    t.datetime  "reset_password_sent_at"
93    t.datetime  "remember_created_at"
94    t.integer  "sign_in_count",  :default => 0,  :null => false
95    t.datetime  "current_sign_in_at"
96    t.datetime  "last_sign_in_at"
97    t.string  "current_sign_in_ip"
98    t.string  "last_sign_in_ip"
99  end
100
101  add_index  "users",  ["email"],  :name => "index_users_on_email",  :unique => true
102  add_index  "users",  ["reset_password_token"],  :name => "index_users_on_reset_password_token",  :unique => true
103
104  end
105```
class CreateUsers < ActiveRecord::Migration
  def change
    create_table :users do |t|
      t.string :first_name
      t.string :last_name
      t.timestamps
    end
  end
end
class AddDeviseToUsers < ActiveRecord::Migration
  def self.up
    change_table(:users) do |t|
      ## Database authenticatable
      t.string :email , null => false, default => ''
      t.string :encrypted_password, null => false, default => ''
      ## Recoverable
      t.string :reset_password_token
      t.datetime :reset_password_sent_at
      ## Rememberable
      t.datetime :remember_created_at
      ## Trackable
      t.integer :sign_in_count, default => 0, null => false
      t.datetime :current_sign_in_at
      t.datetime :last_sign_in_at
      t.string :current_sign_in_ip
      t.string :last_sign_in_ip
      ## Confirmable
      # t.string :confirmation_token
      # t.datetime :confirmed_at
      # t.datetime :confirmation_sent_at
      # t.string :unconfirmed_email # Only if using reconfirmable
      ## Lockable
      # t.integer :failed_attempts, default => 0, null => false # Only if lock strategy is :failed_attempts
      # t.string :unlock_token # Only if unlock strategy is :email or :both
      # t.datetime :locked_at
      # Uncomment below if timestamps were not included in your original model.
      # t.timestamps
    end
    add_index :users, :email , unique => true
    add_index :users, :reset_password_token, unique => true
    # add_index :users, :confirmation_token, unique => true
    # add_index :users, :unlock_token, unique => true
  end

  def self.down
    # By default, we don’t want to make any assumption about how to roll back a migration when your
    # model already existed. Please edit below which fields you would like to remove in this migration.
    raise ActiveRecord::IrreversibleMigration
  end
end
class CreateTweets < ActiveRecord::Migration
  def change
    create_table :my_timeline_twitter_tweets do |t|
      t.datetime :happened_on
      t.text :uri
      t.text :post
      t.references :event
      t.timestamps
    end
  end
end
class CreateMyTimelinePosts < ActiveRecord::Migration
  def change
    create_table :my_timeline_posts do |t|
      t.text :full_text
      t.datetime :happened_on
      t.references :event
      t.timestamps
    end
  end
end
class CreateMyTimelineEvents < ActiveRecord::Migration
  def change
    create_table :my_timeline_events do |t|
      t.text :description
      t.datetime :happened_on
      t.string :icon_name
      t.string :external_link
      t.string :original_id
      t.boolean :public, default: true
      t.integer :importance, default: 5
      t.references :user
      t.references :linkable, polymorphic: true
      t.timestamps
    end
  end
end
# This migration comes from my_timeline (originally 201311030000200)

class CreateMyTimelineSettings < ActiveRecord::Migration

  def change
    create_table :my_timeline_settings do |t|
      t.string :var, :null => false
      t.text :value
      t.references :target, :null => false, :polymorphic => true

      t.timestamps
    end

    add_index :my_timeline_settings, [:target_type, :target_id, :var], :unique => true
  end

end
class CreateGithubForkEvents < ActiveRecord::Migration
  def change
    create_table :my_timeline_github_fork_events do |t|
      t.datetime :happened_on
      t.string :original_id
      t.string :repo
      t.references :event
      t.timestamps
    end
  end
end
class CreateHealthGraphCardioActivities < ActiveRecord::Migration
  def change
    create_table :my_timeline_health_graph_cardio_activities do |t|
      t.datetime :happened_on
      t.float  :meters
      t.float  :duration
      t.integer :calories
      t.string :routefile
      t.string :uri
      t.text   :notes
      t.string :equipment
      t.float  :climb
      t.string :activity_type
      t.references :event
      t.timestamps
    end
  end
end
Hello!
<html lang="en">
  <head>
    <title>Timeline</title>
  </head>

  <body>
    <nav_bar brand: "My Timeline Demo-App", responsive: true do %>
      <%= menu_item "Home", '/', %>
      <%- if current_user %>
        <%= menu_item "My Timeline", '/users/#{current_user.id}/timeline/' %>
        <%= menu_item "Settings", '/users/#{current_user.id}/timeline/control_panel' %>
      <% end %>
    <%- end %>

    <%- menu_divider %>
    <%- dropdown "Theme" do %>
      <%- menu_item "Amelia", '/home/theme/amelia' %>
      <%- menu_item "Cerulean", '/home/theme/cerulean' %>
      <%- menu_item "Cosmo", '/home/theme/cosmo' %>
      <%- menu_item "Cyborg", '/home/theme/cyborg' %>
      <%- menu_item "Flatly", '/home/theme/flatly' %>
      <%- menu_item "Journal", '/home/theme/journal' %>
      <%- menu_item "Readable", '/home/theme/readable' %>
      <%- menu_item "Simplex", '/home/theme/simplex' %>
      <%- menu_item "Slate", '/home/theme/slate' %>
      <%- menu_item "Spacelab", '/home/theme/spacelab' %>
      <%- menu_item "Superhero", '/home/theme/superhero' %>
      <%- menu_item "United", '/home/theme/united' %>
      <%- menu_item "None", '/home/theme/nil' %>
    <% end %>

    <%- menu_item "#{User.first.first_name}'s Timeline", '/users/1/timeline' %>
    <% end %>

    <%- menu_group pull: :right do %>
      <%- if current_user %>
        <%= menu_item "Log Out", destroy_user_session_path, method: :delete %>
        <%= else %>
        <%= menu_item "Log In", new_user_session_path %>
        <%= menu_item "Sign Up", new_user_registration_path %>
      <% end %>
    <% end %>

    <div class="container-fluid">
      <div class="row-fluid">
        <%- bootstrap_flash %>
        <%- yield %>
      </div>
    </div>
  </body>
</html>
<li class="disabled">
  <%= link_to raw(t 'views.pagination.truncate'), '#', %>
</li>
<li class=""active" if page.current? %">1
</li>
<li><%= link_to_unless current_page.last?, raw(t 'views.pagination.last'), url, {:remote => remote} %></li>
<li><%= link_to_unless(current_page).last?, raw(t 'views.pagination.next'), url: rel: 'next', remote: remote %></li>
<%= pagination.render do %>
  <div class="pagination pagination">
    <div>
      <%= first_page_tag unless current_page.first? %>
      <%= prev_page_tag unless current_page.first? %>
      <%= each_page do |page| %>
        <%= page_tag page %>
      <%= end %>
      <%= gap_tag %>
      <%= next_page_tag unless current_page.last? %>
      <%= last_page_tag unless current_page.last? %>
    </div>
  </div>
</%= end %>
/my_timeline-demo/app/views/kaminari/_prev_page.html.erb

1. `<li>`
2. `<%= link_to_unless current_page.first?, raw(t 'views.pagination.previous'), url, rel => 'prev', remote => remote %>`
3. `</li>`
<li><%= link_to_unless current_page.first?, raw(t 'views.pagination.first'), url: :remote => remote %></li>
// This is a manifest file that'll be compiled into application.js, which will include all the files
// listed below.

// Any JavaScript/Coffee file within this directory, lib/assets/javascripts, vendor/assets/javascripts,
// or vendor/assets/javascripts of plugins, if any, can be referenced here using a relative path.

// It's not advisable to add code directly here, but if you do, it'll appear at the bottom of the
// the compiled file.

// WARNING: THE FIRST BLANK LINE MARKS THE END OF WHAT'S TO BE PROCESSED, ANY
// BLANK LINE SHOULD
// GO AFTER THE REQUIRES BELOW.

//= require jquery
//= require jquery_ujs
//= require twitter/bootstrap
//= require_tree .
jQuery ->
$('[rel~=popover], .has-popover').popover()
$('[rel~=tooltip], .has-tooltip').tooltip()
/*
 * This is a manifest file that'll be compiled into application.css, which will include all the files
 * listed below.
 *
 * Any CSS and SCSS file within this directory, lib/assets/stylesheets, vendor/assets/stylesheets,
 * or vendor/assets/stylesheets of plugins, if any, can be referenced here using a relative path.
 *
 * You're free to add application-wide styles to this file and they'll appear at the top of the
 * compiled file, but it's generally better to create a new file per style scope.
 *
 * require_self
 * require_tree .
 */
/my_timeline-demo/app/assets/stylesheets/bootstrap_and_overrides.css

/*
1 require twitter-bootstrap-static/bootstrap
2 require bootswatch/cerulean
3 require twitter-bootstrap-static/sprites
4 =require twitter-bootstrap-static/fontawesome
5 */

select.
6 .date-select
7 {width:auto;
8 border:1px solid #cccccc;
9 background-color:#ffffff;
10 }

select.
11 .datetime-select
12 {width:auto;
13 border:1px solid #cccccc;
14 background-color:#ffffff;
15 }
class User < ActiveRecord::Base
  devise :database_authenticatable, :registerable,
         :recoverable, :rememberable, :trackable, :validatable
  attr_accessible :email, :password, :password_confirmation, :remember_me, :first_name, :last_name

  has_many :events, class_name: MyTimeline::Event
end
module ApplicationHelper
end
class HomeController < ApplicationController
  def index
    end
  def theme
    $current_theme = params[:theme]
    redirect_to :back
    end
  end
end
/my_timeline-demo/app/controllers/application_controller.rb

```ruby
class ApplicationController < ActionController::Base
  # protect_from_forgery
  # skip_before_filter :verify_authentication_token, :only => [:destroy]
end
```
require 'rubygems'

# Set up gems listed in the Gemfile.
ENV['BUNDLE_GEMFILE'] ||= File.expand_path('../../Gemfile', __FILE__)
require 'bundler/setup' if File.exists?(ENV['BUNDLE_GEMFILE'])
TimelineApp::Application.routes.draw do
  devise_for :users
  root to: "home#index"
  get "home/theme/:theme", to: "home#theme", as: :change_theme
  resources :users do
    mount MyTimeline::Engine => '/timeline', as: :my_timeline
  end
end
/my_timeline-demo/config/database.yml

  test:
    adapter: mysql2
    database: timeline_test
    username: root
    password:

  development:
    adapter: mysql2
    database: timeline_dev
    username: root
    password:
/my_timeline-demo/config/application.rb

```ruby
require File.expand_path('../boot', __FILE__)

# Pick the frameworks you want:
require 'active_record/railtie'
require 'action_controller/railtie'
require 'action_mailer/railtie'
require 'active_resource/railtie'
require 'sprockets/railtie'
# require 'rails/test_unit/railtie'

if defined?(Bundler)
  # If you precompile assets before deploying to production, use this line
  Bundler.require('Rails::groups(assets => %w(development test)))
  # If you want your assets lazily compiled in production, use this line
  # Bundler.require(:default, :assets, Rails.env)
end

module TimelineApp
  class Application < Rails::Application
    # Settings in config/environments/* take precedence over those specified here.
    # Application configuration should go into files in config/initializers
    # -- all .rb files in that directory are automatically loaded.
    # Custom directories with classes and modules you want to be autoloadable.
    # config.autoload_paths += %W(#{config.root}/extras)
    # Only load the plugins named here, in the order given (default is alphabetical).
    # .all can be used as a placeholder for all plugins not explicitly named.
    # config.plugins = [ :exception_notification, :ssl_requirement, :all ]
    # Activate observers that should always be running.
    # config.active_record.observers = :cacher, :garbage_collector, :forum_observer
    # Set Time.zone default to the specified zone and make Active Record auto-convert to this zone.
    # Run "rake -D time" for a list of tasks for finding time zone names. Default is UTC.
    # config.time_zone = 'Central Time (US & Canada)'
    # The default locale is :en and all translations from config/locales/*.rb,yml are auto loaded.
    # config:i18n.load_path += Dir[Rails.root.join('my', 'locales', '*.{rb,yml}').to_s]
    # config:i18n.default_locale = :de
    # Configure the default encoding used in templates for Ruby 1.9.
    config.encoding = "utf-8"
    # Configure sensitive parameters which will be filtered from the log file.
    config.filter_parameters += [password]
    # Enable escaping HTML in JSON.
    config.active_support.escape_html_entities_in_json = true
    # Use SQL instead of Active Record's schema dumper when creating the database.
    # This is necessary if your schema can't be completely dumped by the schema dumper,
    # like if you have constraints or database-specific column types
    # config.active_record.schema_format = :sql
    # Enforce whitelist mode for mass assignment.
    # This will create an empty whitelist of attributes available for mass-assignment for all models
    # in your app. As such, your models will need to explicitly whitelist or blacklist accessible
    # parameters by using an attr_accessible or attr_protected declaration.
    config.active_record.whitelist_attributes = true
    # Enable the asset pipeline
    config.assets.enabled = true
    # Version of your assets, change this if you want to expire all your assets
    config.assets.version = '1.0'
  end
end
```
end
/my_timeline-demo/config/environment.rb

1 # Load the rails application
2 require File.expand_path('../application', __FILE__)
3
4 # Initialize the rails application
5 TimelineApp::Application.initialize!
/my_timeline-demo/config/locales/en.yml

# Sample localization file for English. Add more files in this directory for other locales.
# See https://github.com/svenfuchs/rails-i18n/tree/master/rails%2Flocale for starting points.

en:
  hello: "Hello world"
/my_timeline-demo/config/locales/devise.en.yml

# Additional translations at https://github.com/plataformatec/devise/wiki/I18n

en:
devise:
  confirmations:
    confirmed: "Your account was successfully confirmed."
    confirmed_and_signed_in: "Your account was successfully confirmed. You are now signed in"
    send_instructions: "You will receive an email with instructions about how to confirm your account in a few minutes."
    send_paranoid_instructions: "If your email address exists in our database, you will receive an email with instructions about how to confirm your account in a few minutes."
  failure:
    already_authenticated: "You are already signed in."
    inactive: "Your account is not activated yet."
    invalid: "Invalid email or password."
    invalid_token: "Invalid authentication token."
    locked: "Your account is locked."
    not_found_in_database: "Invalid email or password."
    timeout: "Your session expired. Please sign in again to continue."
    unauthenticated: "You need to sign in or sign up before continuing."
    unconfirmed: "You have to confirm your account before continuing."

mailer:
  confirmation_instructions:
    subject: "Confirmation instructions"
  reset_password_instructions:
    subject: "Reset password instructions"
  unlock_instructions:
    subject: "Unlock Instructions"
  omniauth_callbacks:
    failure: "Could not authenticate you from %{kind} because \"%{reason}\"."
    success: "Successfully authenticated from %{kind} account."

passwords:
  no_token: "You can't access this page without coming from a password reset email. If you do come from a password reset email, please make sure you used the full URL provided."
  send_instructions: "You will receive an email with instructions about how to reset your password in a few minutes."
  send_paranoid_instructions: "If your email address exists in our database, you will receive a password recovery link at your email address in a few minutes."
  updated: "Your password was changed successfully. You are now signed in."
  updated_not_active: "Your password was changed successfully."
  registrations:
    destroyed: "Bye! Your account was successfully cancelled. We hope to see you again soon."
    signed_up: "Welcome! You have signed up successfully."
    signed_up_but_inactive: "You have signed up successfully. However, we could not sign you in because your account is not yet activated."
    signed_up_but_locked: "You have signed up successfully. However, we could not sign you in because your account is locked."
    signed_up_but_unconfirmed: "A message with a confirmation link has been sent to your email address. Please open the link to activate your account."
  update_needs_confirmation: "You updated your account successfully, but we need to verify your new email address. Please check your email and click on the confirm link to finalize confirming your new email address."
  updated: "You updated your account successfully."

sessions:
  signed_in: "Signed in successfully."
  signed_out: "Signed out successfully."

unlocks:
  send_instructions: "You will receive an email with instructions about how to unlock your account in a few minutes."
  send_paranoid_instructions: "If your account exists, you will receive an email with instructions about how to unlock it in a few minutes."
  unlocked: "Your account has been unlocked successfully. Please sign in to continue."

errors:
  messages:
    already_confirmed: "was already confirmed, please try signing in"
  confirmation_period_expired: "needs to be confirmed within %{period}, please request a new
one
expired: "has expired, please request a new one"
not_found: "not found"
not_locked: "was not locked"
not_saved: 
one: "1 error prohibited this %{resource} from being saved:"
other: "%{count} errors prohibited this %{resource} from being saved:"
/my_timeline-demo/config/environments/test.rb

```ruby
TimelineApp::Application.configure do
  # Settings specified here will take precedence over those in config/application.rb
  # The test environment is used exclusively to run your application's
test suite. You never need to work with it otherwise. Remember that
your test database is "scratch space" for the test suite and is wiped
and recreated between test runs. Don't rely on the data there!
  config.cache_classes = true

  # Configure static asset server for tests with Cache-Control for performance
  config.serve_static_assets = true
  config.static_cache_control = "public, max-age=3600"

  # Log error messages when you accidentally call methods on nil
  config.whiny_nils = true

  # Show full error reports and disable caching
  config.consider_all_requests_local = true
  config.action_controller.perform_caching = false

  # Raise exceptions instead of rendering exception templates
  config.action_dispatch.show_exceptions = false

  # Disable request forgery protection in test environment
  config.action_controller.allow_forgery_protection = false

  # Tell Action Mailer not to deliver emails to the real world.
  # The :test delivery method accumulates sent emails in the
  # ActionMailer::Base.deliveries array.
  config.action_mailer.delivery_method = :test

  # Raise exception on mass assignment protection for Active Record models
  config.active_record.mass_assignment_sanitizer = :strict

  # Print deprecation notices to the stderr
  config.active_support.deprecation = :stderr

end
```
/my_timeline-demo/config/environments/production.rb

```ruby
TimelineApp::Application.configure do
  # Settings specified here will take precedence over those in config/application.rb
  # Code is not reloaded between requests
  config.cache_classes = true
  # Full error reports are disabled and caching is turned on
  config.consider_all_requests_local = false
  config.action_controller.perform_caching = true

  # Disable Rails's static asset server (Apache or nginx will already do this)
  config.serve_static_assets = false

  # Compress JavaScripts and CSS
  config.assets.compress = true

  # Don't fallback to assets pipeline if a precompiled asset is missed
  config.assets.compile = false

  # Generate digests for assets URLs
  config.assets.digest = true

  # Defaults to nil and saved in location specified by config.assets.prefix
  # config.assets.manifest = YOUR_PATH

  # Specifies the header that your server uses for sending files
  # config.action_dispatch.x_sendfile_header = "X-Sendfile" # for apache
  # config.action_dispatch.x_sendfile_header = 'X-Accel-Redirect' # for nginx

  # Force all access to the app over SSL, use Strict-Transport-Security, and use secure cookies.
  # config.force_ssl = true

  # See everything in the log (default is :info)
  # config.log_level = :debug

  # Prepend all log lines with the following tags
  # config.log_tags = [ :subdomain, :uuid ]

  # Use a different logger for distributed setups
  # config.logger = ActiveSupport::TaggedLogging.new(SyslogLogger.new)

  # Use a different cache store in production
  # config.cache_store = :mem_cache_store

  # Enable serving of images, stylesheets, and JavaScripts from an asset server
  # config.action_controller.asset_host = "http://assets.example.com"

  # Precompile additional assets (application.js, application.css, and all non-JS/CSS are already added
  # config.assets.precompile += %w( search.js )

  # Disable delivery errors, bad email addresses will be ignored
  # config.action_mailer.raise_delivery_errors = false

  # Enable threaded mode
  # config.threadsafe!

  # Enable locale fallbacks for I18n (makes lookups for any locale fall back to
  # the I18n.default_locale when a translation can not be found)
  config.i18n.fallbacks = true

  # Send deprecation notices to registered listeners
  config.active_support.deprecation = :notify

  # Log the query plan for queries taking more than this (works
  # with SQLite, MySQL, and PostgreSQL)
  # config.active_record.auto_explain_threshold_in_seconds = 0.5
end
```
/my_timeline-demo/config/environments/development.rb

```ruby
TimelineApp::Application.configure do
  # Settings specified here will take precedence over those in config/application.rb
  # In the development environment your application's code is reloaded on
  # every request. This slows down response time but is perfect for development
  # since you don't have to restart the web server when you make code changes.
  config.cache_classes = false

  # Log error messages when you accidentally call methods on nil.
  config.whiny_nils = true

  # Show full error reports and disable caching
  config.consider_all_requests_local = true
  config.action_controller.perform_caching = false

  # Don't care if the mailer can't send
  config.action_mailer.raise_delivery_errors = false

  # Print deprecation notices to the Rails logger
  config.active_support.deprecation = :log

  # Only use best-standards-support built into browsers
  config.action_dispatch.best_standards_support = :builtin

  # Raise exception on mass assignment protection for Active Record models
  config.active_record.mass_assignment_sanitizer = :strict

  # Log the query plan for queries taking more than this (works
  # with SQLite, MySQL, and PostgreSQL)
  config.active_record.auto_explain_threshold_in_seconds = 0.5

  # Do not compress assets
  config.assets.compress = false

  # Expands the lines which load the assets
  config.assets.debug = true
end
```
# Use this hook to configure devise mailer, warden hooks and so forth.
# Many of these configuration options can be set straight in your model.

```
Devise.setup do |config|

# The secret key used by Devise. Devise uses this key to generate
# random tokens. Changing this key will render invalid all existing
# confirmation, reset password and unlock tokens in the database.
config.secret_key = '23f2f2b73d75958ca7002d1b3eabf913d273739afbf6d6f75b995bc0396b9b785ed1b1d41daca46e0bdf6f537296f1484449b40bed5ae92ad0dcea5e6bd86c'

# ===> Mailer Configuration
# Configure the e-mail address which will be shown in Devise::Mailer,
# note that it will be overwritten if you use your own mailer class
# with default "from" parameter.
config.mailer_sender = 'please-change-me-at-config-initializers-devise@example.com'

# ===> ORM configuration
# Load and configure the ORM. Supports :active_record (default) and
# :mongoid (bson_ext recommended) by default. Other ORMs may be
# available as additional gems.
require 'devise/orm/active_record'

# ===> Configuration for any authentication mechanism
# Configure which keys are used when authenticating a user. The default is
# just :email. You can configure it to use [:username, :subdomain], so for
# authenticating a user, both parameters are required. Remember that those
# parameters are used only when authenticating and not when retrieving
# session. If you need permissions, you should implement that in a before filter.
# You can also supply a hash where the value is a boolean determining whether
# or not authentication should be aborted when the value is not present.
# config.authentication_keys = [:email]

# Configure parameters from the request object used for authentication. Each entry
# given should be a request method and it will automatically be passed to the
# find_for_authentication method and considered in your model lookup. For instance,
# if you set :request_keys to [:subdomain], :subdomain will be used on authentication.
# The same considerations mentioned for authentication_keys also apply to request_keys.
# config.request_keys = []

# Configure which authentication keys should be case-insensitive.
# These keys will be downcased upon creating or modifying a user and when used
# to authenticate or find a user. Default is :email.
config.case_insensitive_keys = [:email]

# Configure which authentication keys should have whitespace stripped.
# These keys will have whitespace before and after removed upon creating or
# modifying a user and when used to authenticate or find a user. Default is :email.
config.strip_whitespace_keys = [:email]

# Tell if authentication through request.params is enabled. True by default.
# It can be set to an array that will enable params authentication only for the
# given strategies, for example, 'config.params_authenticatable = [:database]' will
# enable it only for database (email + password) authentication.
# config.params_authenticatable = true

# Tell if authentication through HTTP Auth is enabled. False by default.
# It can be set to an array that will enable http authentication only for the
# given strategies, for example, 'config.http_authenticatable = [:token]' will
# enable it only for token authentication. The supported strategies are:
# :database = Support basic authentication with authentication key + password
# :token = Support basic authentication with token authentication key
# :token_options = Support token authentication with options as defined in
# http://api.rubyonrails.org/classes/ActionController/HttpAuthentication/Token.html
# config.http_authenticatable = false
```
/my_timeline-demo/config/initializers/devise.rb
# If http headers should be returned for AJAX requests. True by default.
# config.http_authenticatable_on_xhr = true
# The realm used in Http Basic Authentication. 'Application' by default.
# config.http_authentication_realm = 'Application'
# It will change confirmation, password recovery and other workflows
# to behave the same regardless if the e-mail provided was right or wrong.
# Does not affect registerable.
# config.paranoid = true
# By default Devise will store the user in session. You can skip storage for
# :http_auth and :token_auth by adding those symbols to the array below.
# Notice that if you are skipping storage for all authentication paths, you
# may want to disable generating routes to Devise's sessions controller by
# passing :skip => :sessions to 'devise_for' in your config/routes.rb
config.skip_session_storage = [ :http_auth ]
# By default, Devise cleans up the CSRF token on authentication to
# avoid CSRF token fixation attacks. This means that, when using AJAX
# requests to sign in and sign up, you need to get a new CSRF token
# from the server. You can disable this option at your own risk.
# config.clean_up_csrf_token_on_authentication = true

### Configuration for :database_authenticatable
# For bcrypt, this is the cost for hashing the password and defaults to 10. If
# using other encryptors, it sets how many times you want the password re-encrypted.
# Limiting the stretches to just one in testing will increase the performance of
# your test suite dramatically. However, it is STRONGLY RECOMMENDED not to use
# a value less than 10 in other environments.
config.stretches = Rails.env.test? ? 1 : 10
# Setup a pepper to generate the encrypted password.
# config.pepper = 
# A period that the user is allowed to access the website even without
# confirming his account. For instance, if set to 2.days, the user will be
# able to access the website for two days without confirming his account,
# access will be blocked just in the third day. Default is 0.days, meaning
# the user cannot access the website without confirming his account.
# config.allow_unconfirmed_access_for = 2.days
# A period that the user is allowed to confirm their account before their
# token becomes invalid. For example, if set to 3.days, the user can confirm
# their account within 3 days after the mail was sent, but on the fourth day
# their account can't be confirmed with the token any more.
# Default is nil, meaning there is no restriction on how long a user can take
# before confirming their account.
# config.confirm_within = 3.days
# If true, requires any email changes to be confirmed (exactly the same way as
# initial account confirmation) to be applied. Requires additional unconfirmed_email
# db field (see migrations). Until confirmed new email is stored in
# unconfirmed email column, and copied to email column on successful confirmation.
config.reconfirmable = true
# Defines which key will be used when confirming an account
# config.confirmation_keys = [ :email ]

### Configuration for :confirmable

### Configuration for :rememberable
# The time the user will be remembered without asking for credentials again.
# config.remember_for = 2.weeks
# If true, extends the user's remember period when remembered via cookie.
# config.extend_remember_period = false
# Options to be passed to the created cookie. For instance, you can set
# :secure => true in order to force SSL only cookies.
# config.rememberable_options = {}

# ==> Configuration for :validatable
# Range for password length. Default is 8..128.
config.password_length = 4..128
# Email regex used to validate email formats. It simply asserts that
# one (and only one) @ exists in the given string. This is mainly
# to give user feedback and not to assert the e-mail validity.
# config.email_regexp = /A[^@]+@[^@]+/c/

# ==> Configuration for :timeoutable
# The time you want to timeout the user session without activity. After this
# time the user will be asked for credentials again. Default is 30 minutes.
# config.timeout_in = 30.minutes
# If true, expires auth token on session timeout.
# config.expire_auth_token_on_timeout = false

# ==> Configuration for :lockable
# Defines which strategy will be used to lock an account.
# :failed_attempts = Locks an account after a number of failed attempts to sign in.
# :none = No lock strategy. You should handle locking by yourself.
# config.lock_strategy = :failed_attempts
# Defines which key will be used when locking and unlocking an account
# config.unlock_keys = [:email]
# Defines which strategy will be used to unlock an account.
# :email = Sends an unlock link to the user email
# :time = Re-enables login after a certain amount of time (see :unlock_in below)
# :both = Enables both strategies
# :none = No unlock strategy. You should handle unlocking by yourself.
# config.unlock_strategy = :both
# Number of authentication tries before locking an account if lock_strategy
# is failed attempts.
# config.maximum_attempts = 20
# Time interval to unlock the account if :time is enabled as unlock_strategy.
# config.unlock_in = 1.hour

# ==> Configuration for :recoverable
# Defines which key will be used when recovering the password for an account
# config.reset_password_keys = [:email]
# Time interval you can reset your password with a reset password key.
# Don't put a too small interval or your users won't have the time to
# change their passwords.
config.reset_password_within = 6.hours

# ==> Configuration for :encryptable
# Allow you to use another encryption algorithm besides bcrypt (default). You can use
# :sha1, :sha512 or encryptors from others authentication tools as :clearance_sha1,
# :authlogic_sha512 (then you should set stretches above to 20 for default behavior)
# and :restful_authentication_sha1 (then you should set stretches to 10, and copy
# REST_AUTH_SITE_KEY to pepper).
# Require the `devise-encryptable` gem when using anything other than bcrypt
# config.encryptor = :sha512
# ==> Configuration for :token_authenticatable
# Defines name of the authentication token params key
# config.token_authentication_key = :auth_token

# ==> Scopes configuration
# Turn scoped views on. Before rendering "sessions/new", it will first check for
# "users/sessions/new". It's turned off by default because it's slower if you
# are using only default views.
# config.scoped_views = false

# Configure the default scope given to Warden. By default it's the first
# devise role declared in your routes (usually :user).
# config.default_scope = :user

# Set this configuration to false if you want /users/sign_out to sign out
# only the current scope. By default, Devise signs out all scopes.
# config.sign_out_all_scopes = true

# ==> Navigation configuration
# Lists the formats that should be treated as navigational. Formats like
# :html, should redirect to the sign in page when the user does not have
# access, but formats like :xml or :json, should return 401.
# #
# # If you have any extra navigational formats, like :iphone or :mobile, you
# # should add them to the navigational formats lists.
# #
# # The "/" below is required to match Internet Explorer requests.
# config.navigational_formats = ["/", :html]

# The default HTTP method used to sign out a resource. Default is :delete.
config.sign_out_via = :delete

# ==> OmniAuth
# Add a new OmniAuth provider. Check the wiki for more information on setting
# up on your models and hooks.
# config.omniauth :github, 'APP_ID', 'APP_SECRET', :scope  => 'user,public_repo'

# ==> Warden configuration
# If you want to use other strategies, that are not supported by Devise, or
# change the failure app, you can configure them inside the config.warden block.
# #
# config.warden do |manager|
#   manager.intercept_401 = false
#   manager.default_strategies(:scope => :user).unshift :some_external_strategy
# end

# ==> Mountable engine configurations
# When using Devise inside an engine, let's call it 'MyEngine', and this engine
# is mountable, there are some extra configurations to be taken into account.
# The following options are available, assuming the engine is mounted as:
# #
# mount MyEngine, at: '/my_engine'
# #
# The router that invoked 'devise_for', in the example above, would be:
# config.router_name = 'my_engine'

# When using omniauth, Devise cannot automatically set Omniauth path,
# so you need to do it manually. For the users scope, it would be:
# config.omniauth_path_prefix = '/my_engine/users/auth'

end
# Be sure to restart your server when you modify this file.

# Add new mime types for use in respond_to blocks:

# Mime::Type.register "text/richtext", :rtf

# Mime::Type.register_altas "text/html", :iphone
# Be sure to restart your server when you modify this file.

# Add new inflection rules using the following format
# (all these examples are active by default):
# ActiveSupport::Inflector.inflections do |inflect|
#   inflect.plural /^(ox)$/i, 'oxen'
#   inflect.singular /^(oxen)$/i, 'oxen'
#   inflect.irregular 'person', 'people'
#   inflect.uncountable %w( fish sheep )
# end

# These inflection rules are supported but not enabled by default:
# ActiveSupport::Inflector.inflections do |inflect|
#   inflect.acronym 'RESTful'
# end
MyTimeline.setup do |config|
  # The User class to use... Default is "User".
  # Set to nil to not use per-user timelines,
  # or put a constant in a string to use that class
  config.user_class = "User"
  
  # By default, looks for the user by id, but if you want to use a name or a slug,
  # set it here. I.E., config.user_slug = :nick_name would result in User.find_by_nick_name
  #config.user_slug = :id
  
  # How to render the events - in a :table, or in a :list
  #config.render_method = :table

  # What classes to style the table with
  #config.table_class = "table table-striped"
end

MyTimeline::HealthGraph.setup do |config|
  config.client_id     = "secret"
  config.client_secret = "secret"
end

MyTimeline::Twitter.setup do |config|
  config.consumer_key        = "secret"
  config.consumer_secret     = "secret"
  config.access_token        = "secret"
  config.access_token_secret = "secret"
end

MyTimeline::Github.setup do |config|
  config.client_id     = "secret"
  config.client_secret = "secret"
end
/my_timeline-demo/config/initializers/secret_token.rb

    1  # Be sure to restart your server when you modify this file.
    2  # Your secret key for verifying the integrity of signed cookies.
    3  # If you change this key, all old signed cookies will become invalid!
    4  # Make sure the secret is at least 30 characters and all random,
    5  # no regular words or you'll be exposed to dictionary attacks.
    6  TimelineApp::Application.config.secret_token = '18d5d35e89c7e4fa4e959fd5c60196b83a1cb03d0db7bfead0092f20c9bd7b6332006d7a1c69e4f6d7d41cd5d0aa274d753879a78e70e7304fd4c66114aa'
/my_timeline-demo/config/initializers/session_store.rb

# Be sure to restart your server when you modify this file.

TimelineApp::Application.config.session_store :cookie_store, key: '_timeline_app_session'

# Use the database for sessions instead of the cookie-based default,
# which shouldn’t be used to store highly confidential information
# (create the session table with "rails generate session_migration")
# TimelineApp::Application.config.session_store :active_record_store
/my_timeline-demo/config/initializers/wrap_parameters.rb

1 # Be sure to restart your server when you modify this file.
2 #
3 # This file contains settings for ActionController::ParamsWrapper which
4 # is enabled by default.
5
6 # Enable parameter wrapping for JSON. You can disable this by setting :format to an empty array.
7 ActiveSupport.on_load(:action_controller) do
8   wrap_parameters format: [:json]
9 end
10
11 # Disable root element in JSON by default.
12 ActiveSupport.on_load(:active_record) do
13   self.include_root_in_json = false
14 end
15
/my_timeline-demo/config/initializers/backtrace_silencers.rb

1 # Be sure to restart your server when you modify this file.
2 # You can add backtrace silencers for libraries that you're using but don't wish to see in your backtrace.
3 # Rails.backtrace_cleaner.add_silencer { |line| line =~ /my_noisy_library/ }
4 # You can also remove all the silencers if you're trying to debug a problem that might stem from framework code.
5 # Rails.backtrace_cleaner.remove_silencers!
/my_timeline-demo/public/robots.txt

1 # See http://www.robotstxt.org/wc/norobots.html for documentation on how to use the robots.txt file
2 #
3 # To ban all spiders from the entire site uncomment the next two lines:
4 # User-Agent: *
5 # Disallow: /
6
/my_timeline-demo/script/rails

1  #!/usr/bin/env ruby
2  # This command will automatically be run when you run "rails" with Rails 3 gems installed from the root of your application.
3  
4  APP_PATH = File.expand_path('../..//config/application', __FILE__)
5  require File.expand_path('../..//config/boot', __FILE__)
6  require 'rails/commands'
7  

# My Timeline - Github Plugin

## Requirements:

- [My Timeline](https://github.com/JustinAiken/my_timeline)
- [Octokit](https://github.com/octokit/octokit.rb) gem

## Usage:

1. Add this gem to your Gemfile:
   ```ruby
   `gem 'my_timeline-github'`
   and `bundle install`
   ```
2. [Register your application](https://github.com/settings/applications/new) with Github to get `client_id` and `client_secret` keys.
3. Edit `config/initializers/my_timeline.rb` to include your Github OAUTH keys:
   ```ruby
   MyTimeline.setup do |config|
   ... 
   end
   ```
   ```ruby
   MyTimeline::Github.setup do |config|
   config.client_id     = "YOURKEY"
   config.client_secret = "YOURKEY"
   end
   ```

## Current State

Github provides a wide variety of user activities. This plugin will eventually include them all, but they'll be coming one at a time...

- [CommitCommentEvent](http://developer.github.com/v3/activity/events/types/#commitcommentevent)
- [CreateEvent](http://developer.github.com/v3/activity/events/types/#createevent)
- [DeleteEvent](http://developer.github.com/v3/activity/events/types/#deleteevent)
- [DownloadEvent](http://developer.github.com/v3/activity/events/types/#downloadevent)
- [FollowEvent](http://developer.github.com/v3/activity/events/types/#followevent)
- [ForkEvent](http://developer.github.com/v3/activity/events/types/#forkevent)
- [ForkApplyEvent](http://developer.github.com/v3/activity/events/types/#forkapplyevent)
- [GistEvent](http://developer.github.com/v3/activity/events/types/#gistevent)
- [GollumEvent](http://developer.github.com/v3/activity/events/types/#gollumevent)
- [IssueCommentEvent](http://developer.github.com/v3/activity/events/types/#issuecommentevent)
- [IssuesEvent](http://developer.github.com/v3/activity/events/types/#issuesevent)
- [MemberEvent](http://developer.github.com/v3/activity/events/types/#memberevent)
- [PublicEvent](http://developer.github.com/v3/activity/events/types/#publicevent)
- [PullRequestEvent](http://developer.github.com/v3/activity/events/types/#pullrequestevent)
- [PushEvent](http://developer.github.com/v3/activity/events/types/#pushevent)
- [ReleaseEvent](http://developer.github.com/v3/activity/events/types/#releaserevent)
- [StatusEvent](http://developer.github.com/v3/activity/events/types/#statusevent)
- [TeamAddEvent](http://developer.github.com/v3/activity/events/types/#teamaddevent)
- [WatchEvent](http://developer.github.com/v3/activity/events/types/#watchevent)

## Credits

Original author: [Justin Aiken](https://github.com/JustinAiken)

## Links

- [Source](https://github.com/JustinAiken/my_timeline-github)
## Note on Patches/Pull Requests

* Fork the project.
* Make your feature addition or bug fix.
* Add tests for it. This is important so I don't break it in a future version unintentionally.
* Commit, do not mess with rakefile, version, or history.
* If you want to have your own version, that is fine but bump version in a commit by itself so I can ignore when I pull.
* Send me a pull request. Bonus points for topic branches.

## Copyright

Copyright (c) 2013 Justin Aiken Inc. MIT license (see LICENSE for details).
Copyright 2013 Justin Aiken

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
/my_timeline-github/Gemfile
1  source "https://rubygems.org"
2  gemspec
/my_timeline-github/.gitignore
1  Gemfile.lock
2  .ruby-version
3  .ruby-gemset
4  

# 0.1.0
* Register settings through plugin registry
* Require my_timeline 0.1.0
* Slight refactoring

# 0.0.2
* Added Rails 4 compatibility
* Require my_timeline 0.0.4

# 0.0.1
* Initial release
/my_timeline-github/my_timeline-github.gemspec

```ruby
$:.push File.expand_path(../lib', __FILE__)
require 'my_timeline/github/version'

Gem::Specification.new do |
s.name = "my_timeline-github"
s.version = MyTimeline::Github::VERSION
s.authors = ["Justin Aiken"]
s.email = ["60tonangel@gmail.com"]
s.homepage = "https://www.github.com/JustinAiken/my_timeline-github"
s.summary = "Github plugin for MyTimeline"
s.description = "Github plugin for MyTimeline"
s.license = 'MIT'
s.files = `git ls-files`.split("\n")
s.add_runtime_dependency "my_timeline", '>= 0.1.0'
s.add_runtime_dependency "octokit"
end
```
class CreateGithubCommits < ActiveRecord::Migration
  class CreateGithubCommits < ActiveRecord::Migration
def change
  create_table :my_timeline_github_commits do |t|
  create_table :my_timeline_github_commits do |t|
    t.datetime :happened_on
    t.datetime :happened_on
    t.string :url
    t.string :url
    t.string :sha
    t.string :sha
    t.integer :additions
    t.integer :additions
    t.integer :deletions
    t.integer :deletions
    t.integer :total
    t.integer :total
    t.timestamps
    t.timestamps
  end
  end
  end
end
end
end
/my_timeline-github/db/migrate/create_github_fork_events.rb

class CreateGithubForkEvents < ActiveRecord::Migration
  def change
    create_table :my_timeline_github_fork_events do |t|
      t.datetime :happened_on
      t.string :original_id
      t.string :repo
      t.references :event
      t.timestamps
    end
  end
end
class CreateGithubPushEvents < ActiveRecord::Migration
  def change
    create_table :my_timeline_github_push_events do |t|
      t.datetime :happened_on
      t.string :head
      t.string :ref
      t.integer :size
      t.references :event
      t.timestamps
    end
  end
end
/my_timeline-github/db/migrate/create_github_pull_request_events.rb

```ruby
class CreateGithubPullRequestEvents < ActiveRecord::Migration
  def change
    create_table :my_timeline_github_pull_request_events do |t|
      t.datetime :happened_on
      t.text :title
      t.text :body
      t.string :url
      t.integer :commits
      t.integer :additions
      t.integer :deletions
      t.integer :changed_files
      t.string :repo
      t.references :event
      t.timestamps
    end
  end
end
```
<h3> Github Settings </h3>

<% if @user.settings(:github).user_name %>
  <%= button_to "Scrape activities from #{@user.settings(:github).user_name}!", github_scrape_path, method: :get, class: "btn btn-primary" %>
<% else %>
  <%= form_tag new_github_path, method: :post do %>
    <%= label_tag :user_name %>
    <%= text_field_tag :user_name %>
    <br>
    <%= submit_tag "Save username", class: "btn btn-primary" %>
  <% end %>
<% end %>

<% end %>

/my_timeline-github/app/views/my_timeline/github/_control_panel.html.erb
module MyTimeline
module Github
  class Commit < ActiveRecord::Base
    self.table_name = :my_timeline_github_commits
    belongs_to :push_event, :dependant: :destroy
    attr_protected unless rails4?
  end
end
end
module MyTimeline
module Github
class ForkEvent < ActiveRecord::Base
  self.table_name = :my_timeline_github_fork_events
  belongs_to :event, dependent: :destroy
  attr_protected unless rails4?
end
end
end
module MyTimeline
module Github
class PushEvent < ActiveRecord::Base
  self.table_name = :my_timeline_github_fork_events
  belongs_to :event, dependant: :destroy
  has_many :commits
  attr_protected unless rails4?
end
end
module MyTimeline
  module Github
    class PullRequestEvent < ActiveRecord::Base
      self.table_name = :my_timeline_github_pull_request_events
      belongs_to :event, :dependent => :destroy
      attr_protected unless rails4?
    end
  end
end
module MyTimeline
module Github

class CommitBuilder
  attr_accessor :event

  def build_event(event)
    @event = event
  end

  def build(commit)
    @new_commit = MyTimeline::Github::Commit.create(
      event:      event,
      url:        commit.FIX_THIS,
      sha:        commit.FIX_THIS,
      additions: commit.FIX_THIS,
      deletions:  commit.FIX_THIS,
      total:      commit.FIX_THIS
    )
  end

private
  def foo
    #ADD SHIT HERE
  end
end
end
end
module MyTimeline
  module Github
    class GithubBuilder
      attr_reader :user, :github_event
      def initialize(user, github_event)
        @user = user
        @github_event = github_event
      end
      def build_event
        #Children should define this!
      end
      private
      def already_exists_in_db?
        event = MyTimeline::Event.find_by_original_id(github_event.id)
        event && event.icon_name =~ /github/ 
      end
      def link_user_repo(user_repo)
        return '<a href=https://www.github.com/#{user_repo}>#{user_repo}</a>'
      end
    end
  end
end
module MyTimeline
module Github
class GithubScraper
  # ALL VALID GITHUB API EVENTS:
  #   CommitCommentEvent CreateEvent DeleteEvent DownloadEvent FollowEvent ForkEvent
  #   ForkApplyEvent GistEvent GollumEvent IssueCommentEvent IssuesEvent MemberEvent
  #   PublicEvent PullRequestEvent PullRequestReviewCommentEvent PushEvent ReleaseEvent
  #   StatusEvent TeamAddEvent WatchEvent

  # IMPLEMENTED EVENTS ONLY:
  VALID_EVENTS = %w{ForkEvent PushEvent PullRequestEvent}.freeze

  attr_accessor :user

  def initialize(user)
    @user = user
  end

  def scrape
    @count = 0
    events.each do |event|
      if should_build? event
        @count = @count + 1 if "MyTimeline::Github::#{event.type}::Builder".constantize.new(use
        r.event).build_event
      end
    end
    @count
  end

  private

  def events
    [].tap do |big_array|
      while items_on_current_page? do
        big_array << @current_page
      end
    end.flatten
  end

  def items_on_current_page?
    @current_page = github.user_public_events(username, page: page_number)
    @current_page.length > 0
  end

  def page_number
    @page_number ||= 0
    @page_number = @page_number + 1
  end

  def username
    user.settings(:github).user_name
  end

  def github
    @github ||= Octokit::Client.new(
      client_id: MyTimeline::Github.client_id,
      client_secret: MyTimeline::Github.client_secret,
      auto_traversal: true
    )
  end

  def should_build?(event)
    VALID_EVENTS.include?(event.type) &&
    user #TODO - @user has this option set??
module MyTimeline
  module Github
    class ForkEventBuilder < GithubBuilder
      def build_event
        return false if already_exists_in_db?
        event.linkable = fork_event
        event.user = user if MyTimeline.user_class
        event.save
        fork_event.event = event
        fork_event.save
      end
      private
      def event
        @event ||= MyTimeline::Event.create(
          happened_on: github_event.created_at,
          original_id: github_event.id,
          external_link: "https://www.github.com/#{user.settings(:github).user_name}/#{github_event.repo.name}",
          icon_name: "github.png",
          importance: 5,
          public: 1,
          description: "Forked #{link_user_repo(github_event.repo.name)}"
        )
      end
      def fork_event
        @fork_event ||= MyTimeline::Github::ForkEvent.new(
          happened_on: github_event.created_at,
          original_id: github_event.id,
          repo: github_event.repo.name
        )
      end
    end
  end
end
module MyTimeline
module Github
class PushEventBuilder < GithubBuilder
  def build_event
    return false if already_exists_in_db?
    event.linkable = push_event
    event.user = user if MyTimeline.user_class
    event.save
    push_event.event = event
    push_event.save
    #FIX THIS:
    github_event.commits.each do |commit|
      commit_builder.build commit
    end
  end
  private
  def event
    @event ||= MyTimeline::Event.create(
      happened_on:    github_event.created_at,
      original_id:    github_event.id,
      external_link: github_event.FIXTHIS,
      icon_name:     "github.png",
      importance:    5,
      public:        1,
      description:   "#{github_event.FIXTHIS}" )
  end
  def push_event
    @push_event ||= MyTimeline::Github::PushEvent.new(
       happened_on:    github_event.created_at,
       head:          github_event.FIXTHIS,
       ref:           github_event.FIXTHIS,
       size:          github_event.FIXTHIS
      )
  end
  def commit_builder
    @commit_builder ||= CommitBuilder.new(github_event)
  end
end
end
end
module MyTimeline
  module Github
    class PullRequestEventBuilder < GithubBuilder
      def build_event
        return false if already_exists_in_db?
        event.linkable = pull_request_event
        event.user = user if MyTimeline.user_class
        event.save
        pull_request_event.event = event
        pull_request_event.save
      end
    end
  end
end

private

def event
  @event ||= MyTimeline::Event.create(
    happened_on: happened_on,
    original_id: github_event.id,
    external_link: github_event.payload.pull_request.rels[:html].href,
    icon_name: "github.png",
    importance: 5,
    public: 1,
    description: "#{github_event.payload.action.capitalize} Pull Request #{link_pr} on #{link_user_repo(github_event.repo.name)}: #{github_event.payload.pull_request.title}" )
end

@pull_request_event ||= MyTimeline::Github::PullRequestEvent.new(
  happened_on: happened_on,
  title: github_event.payload.pull_request.title,
  body: github_event.payload.pull_request.body,
  url: github_event.payload.pull_request.rels[:html].href,
  commits: github_event.payload.pull_request.commits,
  additions: github_event.payload.pull_request.additions,
  deletions: github_event.payload.pull_request.deletions,
  changed_files: github_event.payload.pull_request.changed_files,
  repo: github_event.repo.name
)
end

def happened_on
  github_event.created_at
end

def link_pr
  "<a href="#{github_event.payload.pull_request.rels[:html].href}>#{github_event.payload.number}</a>"
end
module MyTimeline
  class GithubController < ApplicationController
    def new
      @user.settings.github.user_name = params[:user_name]
      @user.save!
      redirect_to control_panel_path, notice: "Github added!"
    end
    def scrape
      scrapey = MyTimeline::Github::GithubScraper.new(@user).scrape
      redirect_to :back, notice: "Added #{scrapey} gits."
    end
  end
end
require 'octokit'
require "my_timeline"
require "my_timeline/github/engine"

module MyTimeline
  module Github
    mattr_accessor :client_id, :client_secret
    def self.setup
      yield self
    end
  end
end
/my_timeline-github/lib/my_timeline/github/engine.rb

module MyTimeline
  module Github
    class Engine < ::Rails::Engine
      isolate_namespace MyTimeline::Github
      config.autoload_paths << File.expand_path("../../../app/classes/**", __FILE__)
      config.autoload_paths << File.expand_path("../../../app/scrapers/**", __FILE__)

      config.generators do |g|
        g.test_framework :rspec, fixture: false
        g.fixture_replacement :factory_girl, dir: 'spec/factories'
        g.assets false
        g.helper false
      end

      config.after_initialize do |app|
        MyTimeline.register_plugin :github, defaults: {user_token: nil}
      end

      config.to_prepare do |app|
        MyTimeline.register_plugin :github, defaults: {user_token: nil}
      end

      rake_tasks do
        load File.expand_path "railties/github_tasks.rake", File.dirname(__FILE__)
      end
    end
  end
end
module MyTimeline
  module Github
    VERSION = "0.1.0"
  end
end
namespace "my_timeline-github" do
  def already_copied?(migration_file)
    `ls db/migrate/*#{migration_file}.my_timeline.rb` != ""
  end
  def copy_migration(migration_file)
    return if already_copied? migration_file
    timestamp = Time.now.strftime("%Y%m%d%H%M%S")
    source = File.expand_path "../../../../db/migrate/#{migration_file}.rb", File.dirname(__FILE__)
    dest = File.expand_path "db/migrate/#{timestamp}_#{migration_file}.my_timeline.rb"
    puts "cp #{source} #{dest}"
    `cp #{source} #{dest}`
  end
  namespace :install do
    desc "Copy migrations from my_timeline-github to application"
    task :migrations do
      copy_migration "create_github_fork_events"
      copy_migration "create_github_pull_request_events"
    end
  end
end
/my_timeline-github/config/routes.rb

1  MyTimeline::Engine.routes.draw do
2    get 'github/scrape' => 'github#scrape', as: "github_scrape"
3    post 'github/new' => 'github#new', as: "new_github"
4  resources :github
5  end
6
# My Timeline - Health Graph Plugin

### Health Graph Integration with My Timeline

### Requirements:
- [My Timeline](https://github.com/JustinAiken/my_timeline)
- HealthGraph Client API Key/Secret - Get one [here](http://developer.runkeeper.com/healthgraph/registration-authorization)

### Usage:
1. Add this gem to your Gemfile:
   ```ruby
   gem 'my_timeline-health_graph' and 'bundle install'
   ```
2. Add jupp0r's health_graph library to your Gemfile:
   ```ruby
   gem 'health_graph', git: 'git://github.com/jupp0r/health_graph.git'
   ```
3. Edit `config/initializers/my_timeline.rb` to include your API keys:
   ```ruby
   ```MyTimeline.setup do |config|
   ...
   end
   ```
   ```ruby
   MyTimeline::HealthGraph.setup do |config|
   config.client_id     = "lotsofrandomhexchars"
   config.client_secret = "lotsofrandomhexchars"
   end
   ```

## Credits
Original author: [Justin Aiken](https://github.com/JustinAiken)

## Links
* [Source](https://github.com/JustinAiken/my_timeline-health_graph)
* [Bug Tracker](https://github.com/JustinAiken/my_timeline-health_graph/issues)
* [Rubygem](https://rubygems.org/gems/my_timeline-health_graph)

## Note on Patches/Pull Requests
* Fork the project.
* Make your feature addition or bug fix.
* Add tests for it. This is important so I don't break it in a future version unintentionally.
* Commit, do not mess with rakefile, version, or history.
* If you want to have your own version, that is fine but bump version in a commit by itself so I can ignore when I pull
* Send me a pull request. Bonus points for topic branches.

## Copyright
Copyright (c) 2013 Justin Aiken Inc. MIT license (see LICENSE for details).
Copyright 2013 Justin Aiken

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
# 0.1.0
* Register settings through plugin registry
* Require my_timeline 0.1.0
* Slight refactoring

# 0.0.2
* Added Rails 4 compatibility
* Require my_timeline 0.0.4

# 0.0.1
* Initial release
/my_timeline-health_graph/Gemfile
1  source "https://rubygems.org"
2  gem 'health_graph', git: 'git://github.com/jupp0r/health_graph.git'
3  gemspec
/my_timeline-health_graph/.gitignore

1  Gemfile.lock
2  .ruby-version
3  .ruby-gemset
4  

$:.push File.expand_path("../lib", __FILE__)
require "my_timeline/health_graph/version"

Gem::Specification.new do |
s|
  s.name = "my_timeline-health_graph"
  s.version = MyTimeline::HealthGraph::VERSION
  s.authors = ["Justin Aiken"]
  s.email = ["60tonangel@gmail.com"]
  s.homepage = "https://www.github.com/JustinAiken/my_timeline-health_graph"
  s.summary = "HealthGraph plugin for MyTimeline"
  s.description = "HealthGraph plugin for MyTimeline"
  s.license = 'MIT'
  s.files = `git ls-files`.split("\n")
  s.add_runtime_dependency "my_timeline", '>= 0.1.0'
  s.add_runtime_dependency "health_graph"
end
class CreateHealthGraphCardioActivities < ActiveRecord::Migration
def change
  create_table :my_timeline_health_graph_cardio_activities do |t|
    t.datetime :happened_on
    t.float :meters
    t.float :duration
    t.integer :calories
    t.string :route
    t.string :uri
    t.text :notes
    t.string :equipment
    t.float :climb
    t.string :activity_type
    t.references :event
    t.timestamps
  end
end
<h3> Health Graph Settings </h3>

<% if @user.settings(:health_graph).user_token %>
<%= button_to "Scrape runs!", health_graph_scrape_path, method: :get, class: "btn btn-primary" %>
<% else %>
<%= button_to "Connect to Runkeeper", new_health_graph_path, method: :get, class: "btn btn-primary" %>
<% end %>


module MyTimeline
module HealthGraph
  class CardioActivity < ActiveRecord::Base
    self.table_name = :my_timeline_health_graph_cardio_activities
    belongs_to :event, dependent: :destroy
    attr_protected unless rails4?
    def self.keep_original_time_zone?
      true
    end
  end
end
end
require_relative 'cardio_activity_builder'

module MyTimeline
  module HealthGraph
    class HealthGraphScraper
      attr_accessor :user, :activities, :health_graph_user, :count
      def initialize(user)
        @user = user
        ::HealthGraph.configure do |config|
          config.client_id                  = user.settings(:health_graph).client_id
          config.client_secret               = user.settings(:health_graph).client_secret
          config.authorization_redirect_url = ''
        end
        @health_graph_user = ::HealthGraph::User.new user.settings(:health_graph).user_token
      end
      def scrape(type = :cardio_activities)
        @count = 0
        @activities = []
        case type
          when :cardio_activities then scrape_runs
          when :strength_activities then scrape_strength
        end
        "Added #{count} #{type}
      end
      def scrape_runs
        get_runs_from_rk
        activities.each do |activity|
          @count += 1 if cardio_builder.build_activity(activity)
        end
      end
      def scrape_strength
        #get_strongs_from_rk
        # activities.each do |activity|
        #   @count += 1 if strength_builder.build_activity(activity)
        end
      end
      def get_runs_from_rk
        @feed = health_graph_user.fitness_activities
        orig_feed = @feed.dup
        while @feed
          @activities += @feed.items
          @feed = @feed.next_page
        end
        @activities = orig_feed.send :unpack_items, @activities.reverse
      end
      def cardio_builder
        @cardio_builder ||= MyTimeline::HealthGraph::CardioActivityBuilder.new(user, health_graph_user)
      end
      def strength_builder
        @@strength_builder ||= MyTimeline::HealthGraphScraper::StrengthActivityBuilder.new(user)
      end
    end
  end
end
end
end
end
module MyTimeline
module HealthGraph
class CardioActivityBuilder
  attr_reader :user, :activity, :summary, :health_graph_user
  def initialize(user, health_graph_user)
    @health_graph_user = health_graph_user
    @user = user
  end
  def build_activity(activity)
    @activity = activity
    return false if already_exists_in_db?
    @summary = activity.summary
    event = MyTimeline::Event.create(
      happened_on:    activity.start_time ,
      original_id:    activity.uri[19..26],
      external_link: "#{profile_base_url}/activity/#{activity.uri[19..26]}",
      icon_name: ICONS[activity.type],
      importance:     5,
      public:         true,
      description:    build_description,25
    )
    new_activity = MyTimeline::HealthGraph::CardioActivity.create(28
      happened_on:    activity.start_time ,
      duration:       activity.duration,30
      meters :        activity.total_distance ,31
      uri:            activity.uri ,32
      calories:       summary.total_calories ,33
      climb :         summary.climb,34
      equipment:      summary.equipment ,35
      activity_type: activity.type ,36
      notes:          summary.notes37
    )
    event.linkable = new_activity
    event.user = user if MyTimeline.user_class
    event.save
    new_activity.event = event
    new_activity.save
  end
private
  def already_exists_in_db?
    MyTimeline::HealthGraph::CardioActivity.find_by_uri(@activity.uri.to_s).present?
  end
  def build_description
    """.tap do |s|
      s << ACTIVITY_TEXT[activity.type]
      s << equip_string
      s << to_miles(activity.total_distance)
      s << " in #{to_time(activity.duration)}"
      s << calories_text
      s << notes
    end
  end
  def calories_text
    summary.total_calories ? " and burned #{summary.total_calories.to_i} calories" : ""
end
def notes
  summary.notes ? "(#{summary.notes})" : ""
end

def to_miles(num)
  formatted = "%.2f" % (num * 0.000621371192)  
  "#{formatted} miles"
end

def to_time(num)
  seconds = num % 60  
  minutes = (num / 60) % 60  
  hours = num / (60 * 60)
  format "%02d:%02d:%02d", hours, minutes, seconds
end

ICONS = {
  "Running" => "run.png",
  "Cycling" => "cycle.png",
  "Walking" => "walk.png",
  "Elliptical" => "elliptical.png"
}

ACTIVITY_TEXT = {
  "Running" => "Ran",
  "Cycling" => "Cycled",
  "Walking" => "Strolled",
  "Elliptical" => "Ran"
}

def equip_string
  summary.equipment == "None" ? "" : "(on a #{summary.equipment.downcase})" 
end

def profile_base_url
  health_graph_user.profile.profile
end

def profile_base_url
  health_graph_user.profile.profile
end

def equip_string
  summary.equipment == "None" ? "" : "(on a #{summary.equipment.downcase})" 
end

module MyTimeline

class HealthGraphController < MyTimeline::ApplicationController

  def new
    ::HealthGraph.configure do |config|
      config.client_id = MyTimeline::HealthGraph.client_id
      config.client_secret = MyTimeline::HealthGraph.client_secret
      config.authorization_redirect_url = "#{root_url}/health_graph/code"
    end
    redirect_to ::HealthGraph.authorize_url
  end

  def show
    access_token = ::HealthGraph.access_token(params[:code])
    @user.settings(:health_graph).user_token = access_token
    @user.save!
    redirect_to control_panel_path, notice: "Connection Successful!"
  end

  def scrape
    scrape_status = MyTimeline::HealthGraph::HealthGraphScraper.new(@user).scrape
    redirect_to :back, notice: scrape_status
  end
end
require 'health_graph'
require '"my_timeline"'
require '"my_timeline/health_graph/engine"'

module MyTimeline
  module HealthGraph
    mattr_accessor :client_id, :client_secret

    def self.setup
      yield self
    end
  end
end

end
module MyTimeline
 module HealthGraph
 class Engine < ::Rails::Engine
   isolate_namespace MyTimeline::HealthGraph
   config.autoload_paths << File.expand_path("../../../app/classes/**", __FILE__)
   config.autoload_paths << File.expand_path("../../../app/scrapers/**", __FILE__)
   config.generators do |g|
     g.test_framework :rspec, fixture: false
     g.fixture_replacement :factory_girl, dir: 'spec/factories'
     g.assets false
     g.helper false
   end
   config.after_initialize do |app|
     MyTimeline.register_plugin :health_graph, defaults: {user_token: nil}
   end
   config.to_prepare do |app|
     MyTimeline.register_plugin :health_graph, defaults: {user_token: nil}
   end
   rake_tasks do
     load File.expand_path "railties/health_graph_tasks.rake", File.dirname(__FILE__)
   end
 end
end
end
rake_tasks do
  load File.expand_path "railties/health_graph_tasks.rake", File.dirname(__FILE__)
end
end
end
module MyTimeline
  module HealthGraph
    VERSION = "0.1.0"
  end
end
namespace "my_timeline-health_graph" do
  namespace :install do
    desc "Copy migrations from my_timeline-health_graph to application"
    task :migrations do
      timestamp = Time.now.strftime("%Y%m%d%H%M%S")
      source     = File.expand_path "../../../../db/migrate/20131103010356_create_health_graph_cardio_activities.rb", File.dirname(__FILE__)
      dest       = File.expand_path "db/migrate/#{timestamp}_create_health_graph_cardio_activities.my_timeline.rb"
      puts "cp #{source} #{dest}";
      `cp #{source} #{dest}`
    end
  end
end
/my_timeline-health_graph/config/routes.rb
  1   MyTimeline::Engine.routes.draw do
  2     get 'health_graph/scrape' => 'health_graph#scrape', as: "health_graph_scrape"
  3     resources :health_graph
  4   end
  5
# My Timeline - Twitter Plugin

### Requirements:
- [My Timeline](https://github.com/JustinAiken/my_timeline)
- [Twitter gem](https://github.com/sferik/twitter)

### Usage:
1. Add this gem to your Gemfile:
   ```ruby
   gem 'my_timeline-twitter'
   bundle install
   ```
2. Edit `config/initializers/my_timeline.rb` to include your API keys:
   ```ruby
   MyTimeline.setup do |config|
   ... end
   MyTimeline::Twitter.setup do |config|
   config.client_id = "lotsofrandomhexchars"
   config.client_secret = "lotsofrandomhexchars"
   config.access_token = "lotsofrandomhexchars"
   config.access_token_secret = "lotsofrandomhexchars"
   end
   ```

## Credits
Original author: [Justin Aiken](https://github.com/JustinAiken)

## Links
- [Source](https://github.com/JustinAiken/my_timeline-twitter)
- [Bug Tracker](https://github.com/JustinAiken/my_timeline-twitter/issues)
- [Rubygem](https://rubygems.org/gems/my_timeline-twitter)

## Note on Patches/Pull Requests
- Fork the project.
- Make your feature addition or bug fix.
- Add tests for it. This is important so I don’t break it in a future version unintentionally.
- Commit, do not mess with rakefile, version, or history.
- If you want to have your own version, that is fine but bump version in a commit by itself so I can ignore when I pull
- Send me a pull request. Bonus points for topic branches.

## Copyright
Copyright (c) 2013 Justin Aiken Inc. MIT license (see LICENSE for details).
Copyright 2013 Justin Aiken

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
/my_timeline-twitter/CHANGELOG.md

1  # 0.1.0
2   * Register settings through plugin registry
3   * Require my_timeline 0.1.0
4   * Slight refactoring
5
6  # 0.0.2
7   * Added Rails 4 compatibility
8   * Require my_timeline 0.0.4
9
10 # 0.0.1
11   * Initial release
12
/my_timeline-twitter/Gemfile
1  source "https://rubygems.org"
2
3  gemspec
4
/my_timeline-twitter/.gitignore
1  Gemfile.lock
2  .ruby-version
3  .ruby-gemset
4
/my_timeline-twitter/my_timeline-twitter.gemspec

$:.push File.expand_path("../lib", __FILE__)

require "my_timeline/twitter/version"

Gem::Specification.new do |
  s.name = "my_timeline-twitter"
  s.version = MyTimeline::Twitter::VERSION
  s.authors = ["Justin Aiken"]
  s.email = ["60tonangel@gmail.com"]
  s.homepage = "https://www.github.com/JustinAiken/my_timeline-twitter"
  s.summary = "Twitter plugin for MyTimeline"
  s.description = "Twitter plugin for MyTimeline"
  s.license = 'MIT'
  s.files = `git ls-files`.split("\n")
  s.add_runtime_dependency "my_timeline", '>= 0.1.0'
  s.add_runtime_dependency "twitter", ['>= 5.0', '< 6.0']
end
class CreateTweets < ActiveRecord::Migration
  def change
    create_table :my_timeline_twitter_tweets do |t|
      t.datetime :happened_on
      t.text :uri
      t.text :post
      t.references :event
      t.timestamps
    end
  end
end
<h3> Twitter Settings </h3>

<% if @user.settings(twitter).user_name %>
  <%= button_to "Scrape tweets from #{@user.settings(twitter).user_name}!", twitter_scrape_path, method: :get, class: "btn btn-primary" %>
<% else %>
  <%= form_tag new_twitter_path, method: :post do %>
    <%= label_tag :user_name %>
    <%= text_field_tag :user_name %>
    <br>
    <%= submit_tag "Save username", class: "btn btn-primary" %>
  <% end %>
<% end %>

226
module MyTimeline
  module Twitter
    class Tweet < ActiveRecord::Base
      self.table_name = :my_timeline_twitter_tweets
      belongs_to :event, dependant: :destroy
    end
  end
end
module MyTimeline
module Twitter
class TweetBuilder
  attr_reader :user, :tweet_hash
  def initialize(user)
    @user = user
  end
  def build_tweet(tweet_hash)
    @tweet_hash = tweet_hash
    return false if already_exists_in_db?
    event.linkable = tweet
    event.user = user if MyTimeline.user_class
    event.save
    tweet.event = event
    tweet.save
  end
  private
  def event
    @event ||= MyTimeline::Event.create(
      happened_on:   tweet_hash.created_at, 
      original_id:  tweet_hash.id, 
      external_link: "http://twitter.com/#{user.settings(twitter).user_name}/status/#{tweet_hash.id}, 
      icon_name:     "tweetweet_hash.png", 
      importance:    5, 
      public:        1, 
      description:   linkup_mentions_and_hashtags
    )
  end
  def tweet
    @tweet ||= MyTimeline::Twitter::Tweet.new(
      happened_on:   tweet_hash.created_at, 
      uri:           tweet_hash.id, 
      post:          tweet_hash.text
    )
  end
  def already_exists_in_db?
    MyTimeline::Twitter::Tweet.find_by_uri tweet_hash.id
  end
  def linkup_mentions_and_hashtags
    text = tweet_hash.text.dup
    text.gsub!(/(\[@\w\]+)(\W)?/,
      '<a href="http://twitter.com/\1">@\1</a>\2')
    text.gsub!(/(\#[\w\]+)(\W)?/,
      '<a href="http://twitter.com/search?q=%23\1">#\1</a>\2')
    %Q{"{text}"}
  end
end
end
end
module MyTimeline
  module Twitter
    class TweetScraper
      attr_accessor :user, :tweets
      def initialize(user)
        @user = user
      end
      def scrape
        load_tweets
        @count = 0
        tweets.each do |tweet_hash|
          @count = @count + 1 if builder.build_tweet(tweet_hash)
        end
      end
      def load_tweets
        @tweets = twitter.user_timeline(@user.settings(twitter).user_name, count: 200)
      end
      def builder
        @builder ||= MyTimeline::Twitter::TweetBuilder.new(user)
      end
      def twitter
        @twitter ||= ::Twitter::REST::Client.new do |config|
          config.consumer_key = MyTimeline::Twitter.consumer_key
          config.consumer_secret = MyTimeline::Twitter.consumer_secret
          config.access_token = MyTimeline::Twitter.access_token
          config.access_token_secret = MyTimeline::Twitter.access_token_secret
        end
      end
    end
  end
end

module MyTimeline

class TwitterController < ApplicationController

  def new
    @user.settings(twitter).user_name = params[:user_name]
    @user.save!
    redirect_to control_panel_path, notice: "Twitter added!"
  end

def scrape
    scrapey = MyTimeline::Twitter::TweetScraper.new(@user).scrape
    redirect_to :back, notice: "Added #{scrapey} tweets."
  end
end
end
require 'twitter'
require "my_timeline"
require "my_timeline/twitter/engine"

module MyTimeline
  module Twitter
    mattr_accessor :consumer_key, :consumer_secret, :access_token, :access_token_secret
    def self.setup
      yield self
    end
  end
end
module MyTimeline
  module Twitter
    class Engine < ::Rails::Engine
      isolate_namespace MyTimeline::Twitter
      config.autoload_paths << File.expand_path("../../../app/classes/**", __FILE__)
      config.autoload_paths << File.expand_path("../../../app/scrapers/**", __FILE__)
      config.generators do |g|
        g.test_framework :rspec, fixture: false
        g.fixture_replacement :factory_girl, dir: "spec/factories"
        g.assets false
        g.helper false
      end
      config.after_initialize do |app|
        MyTimeline.register_plugin :twitter, defaults: { user_token: nil }
      end
      config.to_prepare do |app|
        MyTimeline.register_plugin :twitter, defaults: { user_token: nil }
      end
      rake_tasks do
        load File.expand_path "railties/twitter_tasks.rake", File.dirname(__FILE__)
      end
    end
  end
end
module MyTimeline
module Twitter
  VERSION = "0.1.0"
end
end
namespace "my_timeline-twitter" do
  namespace :install do
    desc "Copy migrations from my_timeline-twitter to application"
    task :migrations do
      timestamp = Time.now.strftime("%Y%m%d%H%M%S")
      source     = File.expand_path("../../../../db/migrate/20131107025006_create_tweets.rb", __FILE__)
      dest       = File.expand_path("db/migrate/\#{timestamp}_create_tweets.my_timeline.rb")
      puts "cp #{source} #{dest}"
      ` cp #{source} #{dest}`
    end
  end
end
/my_timeline-twitter/config/routes.rb

```
MyTimeline::Engine.routes.draw do
  get 'twitter/scrape' => 'twitter#scrape', as: "twitter_scrape"
  post 'twitter/new' => 'twitter#new', as: "new_twitter"
  resources :twitter

end
```