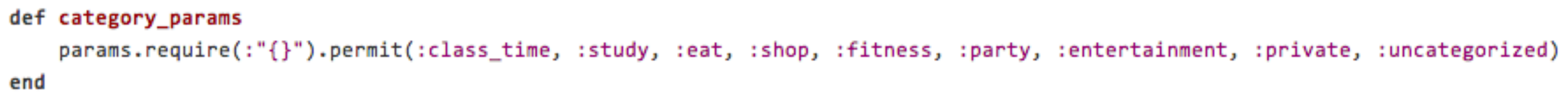
Synapse

Team Friends!

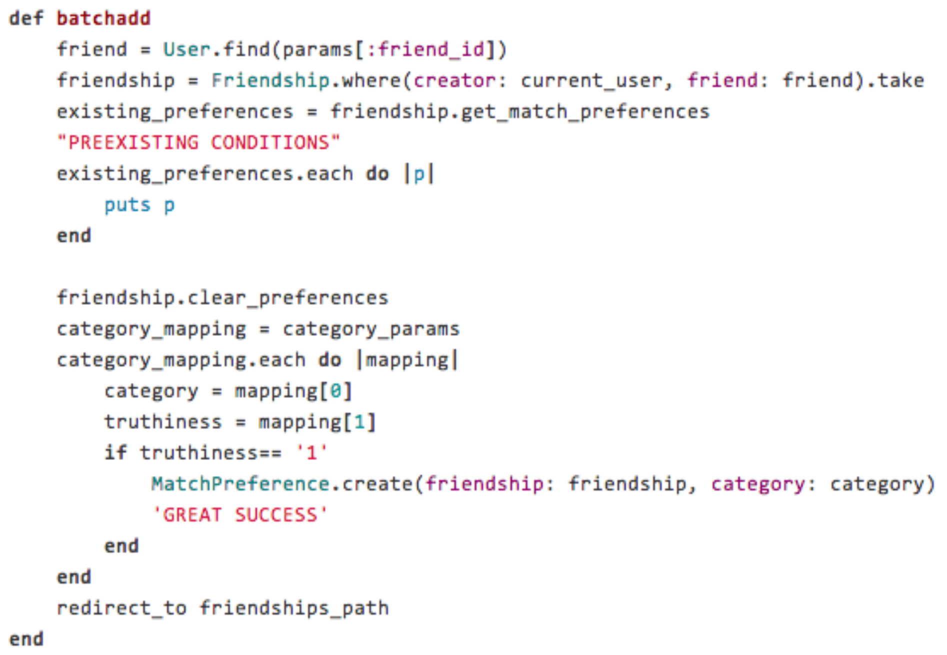
Code Critique by String Quartet

Passing category params

Ido Efrati



```ruby
def category_params
    params.require(:"{}").permit(:class_time, :study, :eat, :shop, :fitness, :party, :entertainment, :private, :uncategorized)
end
```



```ruby
def batchadd
    friend = User.find(params[:friend_id])
    friendship = Friendship.where(creator: current_user, friend: friend).take
    existing_preferences = friendship.get_match_preferences
    "PREEXISTING CONDITIONS"
    existing_preferences.each do |p|
        puts p
    end

    friendship.clear_preferences
    category_mapping = category_params
    category_mapping.each do |mapping|
        category = mapping[0]
        truthiness = mapping[1]
        if truthiness== '1'
            MatchPreference.create(friendship: friendship, category: category)
            'GREAT SUCCESS'
        end
    end
    redirect_to friendships_path
end
```

The intention is to check whether a category was selected or not. Each param is mapped to a checkbox in the view, and if a checkbox was marked a match will be created.

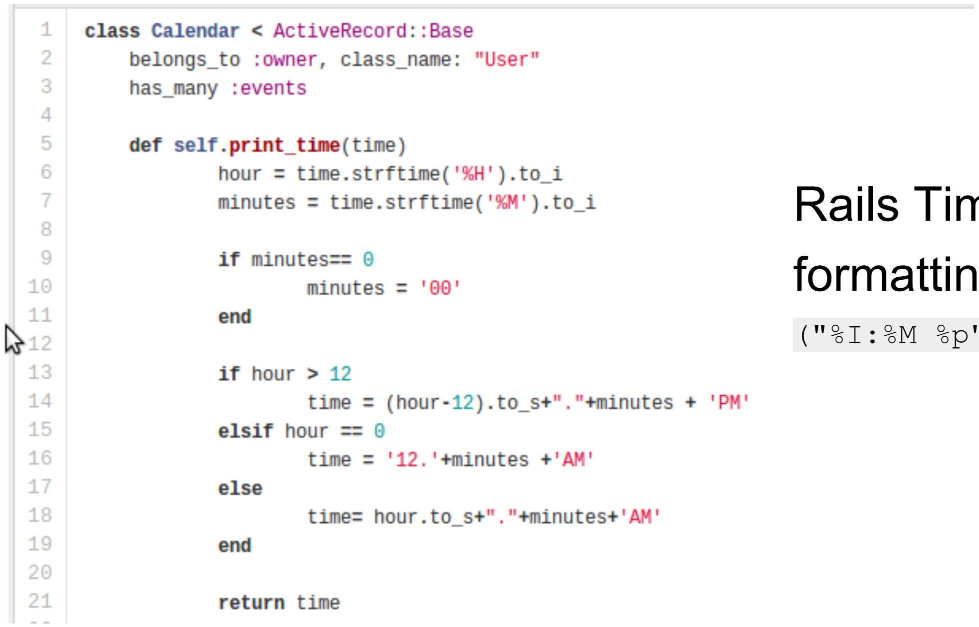
However, this will not be efficient if you would like to add more categories, or if you would like to let your users add new categories.

First you should consider adding a table for categories so users will be able to create new tags. Second, consider changing your strong params to pass an array. To declare that the value in params must be an array of permitted scalar values map the key to an empty array:

```ruby
params.permit(:id => [])
```

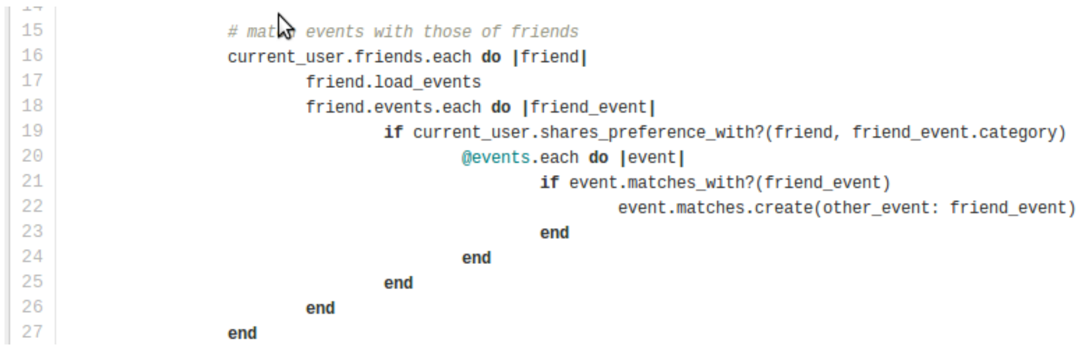
*https://github.com/rails/strong_parameters

Cynthia Jing



```ruby
class Calendar < ActiveRecord::Base
    belongs_to :owner, class_name: "User"
    has_many :events

    def self.print_time(time)
            hour = time.strftime('%H').to_i
            minutes = time.strftime('%M').to_i

            if minutes== 0
                    minutes = '00'
            end

            if hour > 12
                    time = (hour-12).to_s+"."+minutes + 'PM'
            elsif hour == 0
                    time = '12.'+minutes +'AM'
            else
                    time= hour.to_s+"."+minutes+'AM'
            end

            return time
```

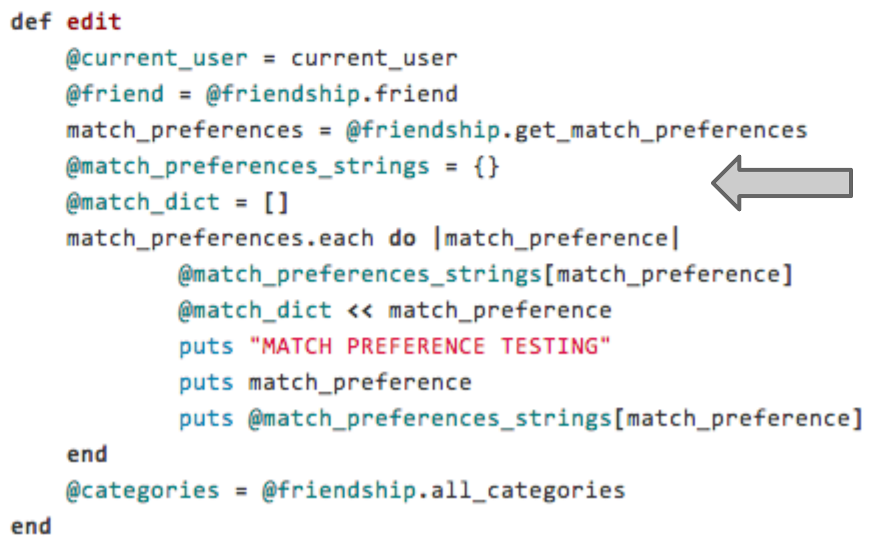
Rails Time and DateTime have formatting defined already, ie `strftime("%I:%M %p")` = output: HH:MM AM



```ruby
# match events with those of friends
current_user.friends.each do |friend|
        friend.load_events
        friend.events.each do |friend_event|
                if current_user.shares_preference_with?(friend, friend_event.category)
                        @events.each do |event|
                                if event.matches_with?(friend_event)
                                        event.matches.create(other_event: friend_event)
                                end
                        end
                end
        end
end
```

You should use a model method to at least return all the events that match with the current user

Carrie Cai



```ruby
def edit
    @current_user = current_user
    @friend = @friendship.friend
    match_preferences = @friendship.get_match_preferences
    @match_preferences_strings = {}
    @match_dict = []
    match_preferences.each do |match_preference|
            @match_preferences_strings[match_preference]
            @match_dict << match_preference
            puts "MATCH PREFERENCE TESTING"
            puts match_preference
            puts @match_preferences_strings[match_preference]
    end
    @categories = @friendship.all_categories
end
```

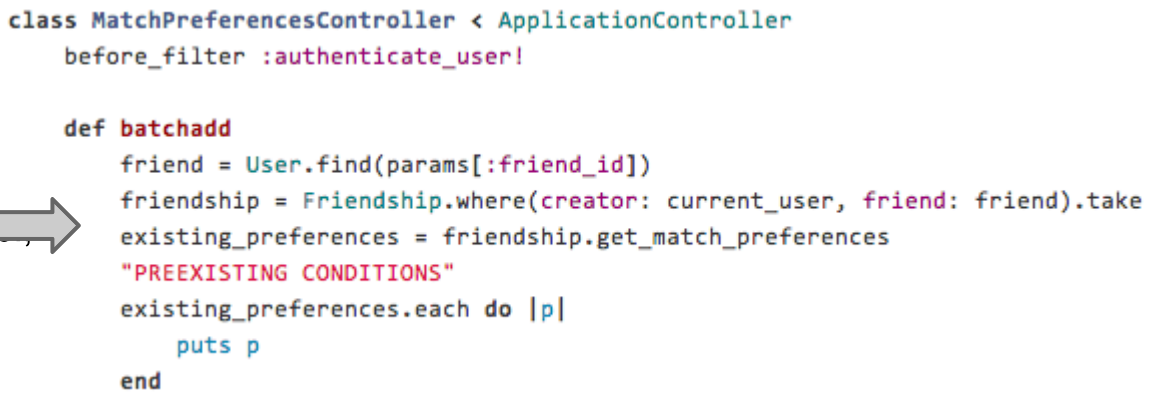
FriendshipController:

This method could use some comments, or alternatively, better variable names.

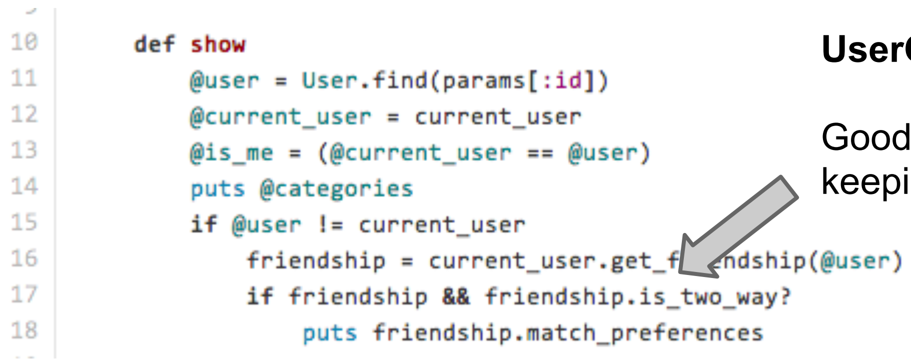
For example, it is unclear at first glance why it is necessary to have both @match_preferences_strings and @match_dict.

MatchPreferencesController:

The logic in lines 2 and 3 of this method should be merged and moved to the model, and called in the controller using a more conceptual method name (i.e. “get preferences”).



```ruby
class MatchPreferencesController < ApplicationController
    before_filter :authenticate_user!

    def batchadd
        friend = User.find(params[:friend_id])
        friendship = Friendship.where(creator: current_user, friend: friend).take
        existing_preferences = friendship.get_match_preferences
        "PREEXISTING CONDITIONS"
        existing_preferences.each do |p|
            puts p
        end
```



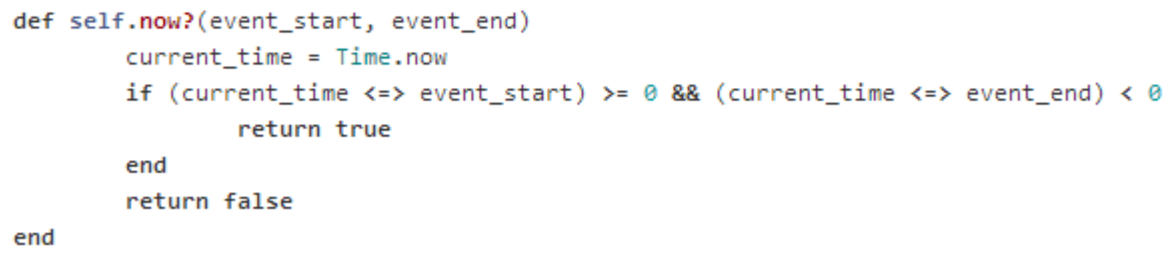
```ruby
    def show
        @user = User.find(params[:id])
        @current_user = current_user
        @is_me = (@current_user == @user)
        puts @categories
        if @user != current_user
            friendship = current_user.get_friendship(@user)
            if friendship && friendship.is_two_way?
                puts friendship.match_preferences
```

UserController:

Good job putting logic in the model and keeping the controller skinny!

Nitya Subramanian

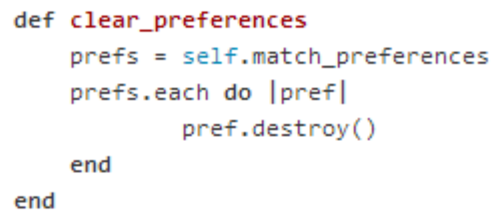
Calendar Model



```
def self.now?(event_start, event_end)
        current_time = Time.now
        if (current_time <=> event_start) >= 0 && (current_time <=> event_end) < 0
                return true
        end
        return false
end
```

Consider using the 'cover' function here: (event_start..event_end).cover?(Time.now) instead of this method(this function takes in two timestamps and will avoid any strange edge cases with timezones/strange dates

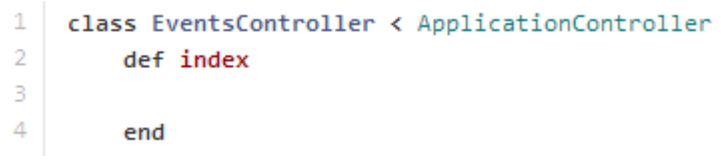
Friendship Model



```
def clear_preferences
    prefs = self.match_preferences
    prefs.each do |pref|
            pref.destroy()
    end
end
```

You could instead use self.
match_preferences.destroy_all

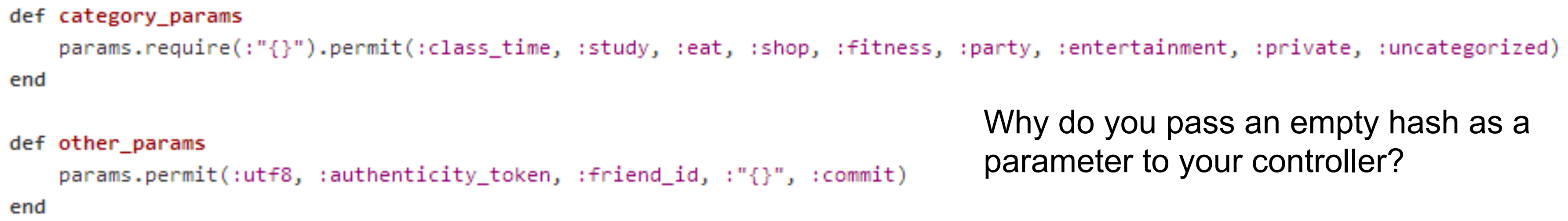
Events Controller



```
class EventsController < ApplicationController
    def index

    end
```

This controller probably isn't necessary, and should be removed

Match Preferences Controller



```
def category_params
    params.require(:"{}").permit(:class_time, :study, :eat, :shop, :fitness, :party, :entertainment, :private, :uncategorized)
end

def other_params
    params.permit(:utf8, :authenticity_token, :friend_id, :"{}", :commit)
end
```

Why do you pass an empty hash as a parameter to your controller?