JIRA Report: time worked per user, per sprint

JIRA REST-based Reporting Scripts
Scripts (mostly written in Ruby) generating reports from a remote JIRA, via JIRA's REST API.

Report Synopsis

- Given a sprint, print the total worklog hours logged by each user on the sprint's issues.
- Given a sprint, report total worklog hours.
- Given a sprint, report total Story Points completed.

Implementation

The scripts achieving these goals are found in Bitbucket at https://bitbucket.org/redradish/jira-ruby-reports/src/master/time_spent_per_sprint/?at=master. Sample use:

```
jturner@jturner-desktop ~/src/bitbucket.org/redradish/jira-ruby-reports/time_spent_per_sprint $ bundle exec .
/time_per_user_per_sprint.rb "sprint=Quintara"
jsmith: 32h 16m
afernando: 29h 30m
jalison: 37h 12m
oporter: 44h 40m
dnewlands: 32h 30m
ballen: 0h 1m
bob: 12h 31m

jturner@jturner-desktop ~/src/bitbucket.org/redradish/jira-ruby-reports/time_spent_per_sprint $ bundle exec .
/sprint_total_time.rb
Total time spent:       224.000000
Total Story Points resolved:    61
```

Implementation Walkthrough

As usual, we first create a `$client` to query JIRA with:

```
require 'jira'
#require 'pry'
require 'parallel'
require_relative './vars'
$options = {
  :site => JIRA_URL,
  :context_path => '',
  :username => JIRA_USERNAME,
  :password => JIRA_PASSWORD,
  :auth_type => :basic
}
query=ARGV[0]
if !query then
        query="sprint=251"
        $stderr.puts "No JQL argument passed; using default: #{query}"
end
$client = JIRA::Client.new($options)
```

and fetch relevant issues with JQL:

```
$client.Issue.jql("sprint=251", max_results:1000)
```

We are interested in the worklogs, however, and worklogs are not fetched by default by the Ruby library. We need to call `.fetch` on each issue to fetch the `worklogs` field.

Fetching details of hundreds of issues is going to take time, so we use Ruby's `parallel` Gem to parallelize this fetching somewhat:

```
issues = Parallel.map($client.Issue.jql("sprint=251", max_results:1000), :in_processes=>10) { |i| i.fetch; i }
```

The rest of the script builds on this, building up a hash mapping authornames to the cumulative total of their worklogs, and then printing the hash:

```
issues = Parallel.map($client.Issue.jql("sprint=251", max_results:1000), :in_processes=>10) { |i| i.fetch; i }
  .each_with_object({}) { |i,hash|
        i.worklogs.each { |w|
            hash[w.author.name] ||= 0
            hash[w.author.name] += w.timeSpentSeconds
        } }
    .each { |user, secs|
        puts "#{user}: #{secs / 60 / 60}h #{secs / 60 % 60}m"
    }
```

The `sprint_total_time.rb` script is even simpler:

```
issues = Parallel.map($client.Issue.jql("sprint=251", max_results:1000), :in_processes=>10) { |i| i.fetch; i }
puts "Total time spent:\t%f\n" % (issues.collect { |i| i.timespent or 0 }.inject(:+) / 60 / 60)
puts "Total Story Points resolved:\t%d\n" % issues.collect { |i| i.customfield_10105 or 0 }.inject(:+)
```