# JIRA Report: overdue/on-time issue counts per priority

**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 JIRA project name, a start date and and end date, find total counts of issues completed before, on or after the due date, per priority.

Generally the issues in question will be bug reports. This report answers questions like "how many bugs of each priority are we resolving each month?" or "Are we resolving bugs by their due date"? By running the report for successive time periods, one can get a feel for the total backlog, and per-month progress to clearing the backlog.Sample use: for Bugs in the UX project, show how many were resolved before, on or after their Due date, in the month of July:

```
jturner@jturner-desktop ~/src/bitbucket.org/redradish/jira-ruby-reports/overdue_by_priority $ bundle exec .
/jira_overdue_by_priority_report.rb --project=UX --issuetype=Bug --from=2015-07-01 --to=2015-07-30
-------------------------------------------------
|                                | Total | Unfinished | Finished Before Due | Finished On Due | Finished
After Due | Finished, no due date | Finished outside daterange |
| Major, without due date           | 328   | 6          | 0                   | 0               |
0                 | 24                    | 298                         |
| Minor, without due date           | 125   | 83         | 0                   | 0               |
0                 | 3                     | 39                          |
| Blocker, without due date         | 9     | 0          | 0                   | 0               |
0                 | 0                     | 9                           |
| Critical, without due date        | 14    | 0          | 0                   | 0               |
0                 | 0                     | 14                          |
| Minor, with due date              | 1     | 1          | 0                   | 0               |
0                 | 0                     | 0                           |
| Major, with due date              | 2     | 2          | 0                   | 0               |
0                 | 0                     | 0                           |
| Unplanned, with due date          | 1     | 0          | 0                   | 0               |
0                 | 0                     | 1                           |
| Unplanned, without due date       | 21    | 0          | 0                   | 0               |
0                 | 0                     | 21                          |
-------------------------------------------------
<table border=1>
<tr><th>    </th><th>Total</th><th>Unfinished</th><th>Finished Before Due</th><th>Finished On Due<
/th><th>Finished After Due</th><th>Finished, no due date</th><th>Finished outside daterange</th></tr>
<tr><th>Major, without due date</th><td>328</td><td>6</td><td>0</td><td>0</td><td>0</td><td>24</td><td>298</td><
/tr>
<tr><th>Minor, without due date</th><td>125</td><td>83</td><td>0</td><td>0</td><td>0</td><td>3</td><td>39</td><
/tr>
<tr><th>Blocker, without due date</th><td>9</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>9</td><
/tr>
<tr><th>Critical, without due date</th><td>14</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>14</td><
/tr>
<tr><th>Minor, with due date</th><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><th>Major, with due date</th><td>2</td><td>2</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><th>Unplanned, with due date</th><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td></tr>
<tr><th>Unplanned, without due date</th><td>21</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>21<
/td></tr></table>
```

The script's output, when rendered:

| | Total | Unfinished | Finished Before Due | Finished On Due | Finished After Due | Finished, no due date | Finished outside daterange |
|---|---|---|---|---|---|---|---|
| **Major, without due date** | 328 | 6 | 0 | 0 | 0 | 24 | 298 |
| **Minor, without due date** | 125 | 83 | 0 | 0 | 0 | 3 | 39 |
| **Blocker, without due date** | 9 | 0 | 0 | 0 | 0 | 0 | 9 |
| **Critical, without due date** | 14 | 0 | 0 | 0 | 0 | 0 | 14 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Minor, with due date** | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| **Major, with due date** | 2 | 2 | 0 | 0 | 0 | 0 | 0 |
| **Unplanned, with due date** | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| **Unplanned, without due date** | 21 | 0 | 0 | 0 | 0 | 0 | 21 |

Here we can see:

- **Total** is the total issue count (for that row). In the example, at the time we ran the script there were **328** Bugs with 'Major' priority and no due date.
- **Unfinished** is the total unresolved issue count. In the example, of the 328 Major unscheduled bugs, there were **6** unresolved.
- The next 4 cols, **Finished Before/On/After Due,** and **Finished, no due date**, show issues resolved within the given interval. In the example, **24** of the 328 bugs were resolved in September.
- The last column, **Finished outside daterange**, shows issues resolved outside the given interval. In the example, **298** of the 328 bugs were resolved outside September.

# Implementation

The script achieving this is found in Bitbucket at https://bitbucket.org/redradish/jira-ruby-reports/src/master/overdue_by_priority/.

## Implementation Walkthrough

In Ruby, using the `jira-ruby` gem.

First we set up a `$client` object, using HTTP Basic authentication;

```
require 'jira'
require 'parallel'

HOST='https://REDACTED.atlassian.net'
$options = {
  :site => HOST,
  :context_path => '',
  :username => 'myusername',
  :password => %q{REDACTED},
  :auth_type => :basic
}

$client = JIRA::Client.new($options)
```

Next, we fetch the issues we're interested in:

```
issues = $client.Issue.jql("project=UX and updated>='2015-10-01' AND updated<='2015-10-27'", max_results:1000)
{ |i| i.fetch; i }
```

Now for the interesting part. Issues with a due date will have a `resolutiondate` field, which we can parse wtih `strptime`:

```
rdate = issues.find { |i| i.resolutiondate  }.resolutiondate
=> "2015-10-26T09:23:07.000-0700"
rdate = DateTime.strptime(rdate, '%Y-%m-%dT%H:%M:%S.%L%z')
=> #<DateTime: 2015-10-26T09:23:07-07:00 ((2457322j,58987s,0n),-25200s,2299161j)>
rdate = rdate.to_date                # Discard time portion
=> #<Date: 2015-10-26 ((2457322j,0s,0n),+0s,2299161j)>
```

We will also have a `duedate`, which we can parse similarly:

```
ddate = issues.find { |i| i.duedate  }.duedate
=> "2015-11-05"
Date.strptime(ddate, "%Y-%m-%d")
=> #<Date: 2015-11-05 ((2457332j,0s,0n),+0s,2299161j)>
```

and a `priority`, which is actually an object, so we'll just use the `name` part of it:

```
[14] pry(main)> ddate = issues.find { |i| i.priority  }.priority.name
=> "Critical"
```

Now we need to:

- group issues by priority
  - for each priority's group, group again by classification:
    - if there is no resolution date, classify as "Unfinished"
    - if there is a resolution date, but no due date, classify "Finished, no due date"
    - If the resolution date and due date match, classify as "On Due"
    - If the resolution date is earlier than due date, classify as "Before Due"
    - If the resolution date is after the due date, classify as "After Due"

The Ruby `Enumerable` module's `group_by` method does the group-into-buckets job nicely, giving us a hash-of-hashes data structure.

```
data = issues.group_by { |i|
                i.priority.name + ", " + (i.duedate ? "with" : "without") + " due date" }
        .inject({}) { |h, (priority, issues)|
                h[priority] = issues.group_by { |i|
                        resdate = i.resolutiondate && DateTime.strptime(i.resolutiondate, '%Y-%m-%dT%H:%M:%S.%L%
z').to_date
                        duedate = i.duedate && Date.strptime(i.duedate, "%Y-%m-%d")
                        if !resdate then "Unfinished"
                                elsif !duedate then "Finished, no due date"
                                elsif resdate == duedate then "Finished On Due"
                                elsif resdate < duedate then "Finished Before Due"
                                else "Finished After Due"
                        end
                }
                h[priority]["Total"] = issues
                h
         }

data.keys   # Show our top-level groupings (this will be rows)
=> ["Critical, without due date", "Major, without due date", "Minor, without due date", "Major, with due date",
"Blocker, without due date"]
cols = data.collect { |(k,v)| v.keys  }.flatten.uniq  # Identify unique columns.
=> ["Unfinished", "Finished, no due date"]
```

## Reporting

We now have our data in a nested-hash data structure, and want to output it in tabular format.

First, we iterate over rows and columns and count the issues, giving us a simple 2d structure:

```
cols = ["Total", "Unfinished", "Finished On Due", "Finished Before Due", "Finished After Due", "Finished, no
due date"]
result = [[nil] + cols] # First row is a list of columns, starting with a nil
# Add rows, consisting of an array beginning with 'rowname', followed by the number of issues, or zero
result += data.collect { |(priority, issues_by_finishedstatus)|
        [priority] + cols.collect { |col|
                 issues_by_finishedstatus[col] ? issues_by_finishedstatus[col].size : 0 }
        }
=> pp result
[[nil,
  "Total",
  "Unfinished",
  "Finished On Due",
  "Finished Before Due",
  "Finished After Due",
  "Finished, no due date"],
 ["Minor, without due date", 44, 36, 0, 0, 0, 8],
 ["Blocker, without due date", 5, 0, 0, 0, 0, 5],
 ["Critical, without due date", 4, 1, 0, 0, 0, 3],
 ["Major, without due date", 131, 46, 0, 0, 0, 85],
 ["Minor, with due date", 1, 1, 0, 0, 0, 0],
 ["Major, with due date", 2, 2, 0, 0, 0, 0],
 ["Blocker, with due date", 1, 1, 0, 0, 0, 0]]
```

Displaying our array-of-arrays properly indented can be done with:

```
puts "| " + result.collect { |r| r.collect.with_index { |c,i|
                colwidth = (i==0 ? 26 : result[0][i].size)
                "%-#{colwidth}s" % c }.join(" | ")
        }.join(" |\n| ") + " |"
=>
|                           | Total | Unfinished | Finished On Due | Finished Before Due | Finished After Due
| Finished, no due date |
| Minor, without due date    | 44    | 36         | 0               | 0                   | 0
| 8                     |
| Blocker, without due date  | 5     | 0          | 0               | 0                   | 0
| 5                     |
| Critical, without due date | 4     | 1          | 0               | 0                   | 0
| 3                     |
| Major, without due date    | 131   | 46         | 0               | 0                   | 0
| 85                    |
| Minor, with due date       | 1     | 1          | 0               | 0                   | 0
| 0                     |
| Major, with due date       | 2     | 2          | 0               | 0                   | 0
| 0                     |
| Blocker, with due date     | 1     | 1          | 0               | 0                   | 0
| 0                     |
```

The script in Bitbucket also emits HTML, which renders as:

| | Total | Unfinished | Finished On Due | Finished Before Due | Finished After Due | Finished, no due date |
|---|---|---|---|---|---|---|
| **Minor, without due date** | 44 | 36 | 0 | 0 | 0 | 8 |
| **Blocker, without due date** | 5 | 0 | 0 | 0 | 0 | 5 |
| **Critical, without due date** | 4 | 1 | 0 | 0 | 0 | 3 |
| **Major, without due date** | 131 | 46 | 0 | 0 | 0 | 85 |
| **Minor, with due date** | 1 | 1 | 0 | 0 | 0 | 0 |
| **Major, with due date** | 2 | 2 | 0 | 0 | 0 | 0 |
| **Blocker, with due date** | 1 | 1 | 0 | 0 | 0 | 0 |