Worklog Simple Group By Report

This report displays grouped worklogs as a table with the possibility to expand/collapse rows by categories.

Report preview

Time Spent 2 weeks, 4 days, 6 hours

Issue Type	2017 - 01	2017 - 02
IssueType 5	1 day	0 minutes
Caro B	3 hours	0 minutes
admin	5 hours	0 minutes
Task	2 hours	2 weeks, 3 days, 4 hours
admin	2 hours	2 weeks, 3 days, 4 hours
Σ	**1 day, 2 hours**	**2 weeks, 3 days, 4 hours**

Parameters

Parameter		Type	Default value
UI	Code		
JQL	`JQL`	JQL Autocomplete	
Group By	`Grouping`	Group By Picker	Issue Type
Time Period	`TimePeriod`	Time Period Picker	Month
From	`From`	Interval Picker	-1m
To	`To`	Interval Picker	+1m

Layout Script

Used layout: **Extended Worklog**.

```
var div = $("#chart");
var what;

if (chartData.sums.length < 2) {
    $(div).append("<h4>" + chartData.empty + "</h4>");
}
else {
    if (chartData.title) {
        $(div).append("<h4>" + chartData.title + " <small>" + chartData.sum + "</small></h4>");
    }
    var table = d3.select("#chart").append('table')
    var thead = table.append('thead')
    var tbody = table.append('tbody');
```

```
    $("#chart > table").addClass("table")
    $("#chart > table").addClass("table-hover")

    thead.append('tr').selectAll('th').data(chartData.labels).enter().append('th').text(function(column) {
        return column;
    });

    var expanded = true;

    $('thead').on("click", function()
    {
        var childs = $("tbody").children();
        if (expanded)
        {
            $.each(childs, function(index, element) {
                if (!$(element).hasClass("active")) {
                    $(element).hide();
                }
            });
        }
        else
        {
            $.each(childs, function(index, element) {
                if (!$(element).hasClass("active")) {
                    $(element).show();
                }
            });
        }
        expanded = !expanded;
    });

    $('thead > tr').css("cursor", "pointer");

    var tr = tbody.selectAll('tr').data(chartData.columns).enter().append("tr").classed("active", function
(value, index)
    {
        return value.active;
    }).style("font-weight", function(value, index) {
        return value.fontweight;
    });

    var td = tr.selectAll("td").data(function(tr_data,index) {
        return chartData.sums[index];
    }).enter().append("td").text(function(d) {
        return d;
    });

    $.each($(".active:not(:last-child)"), function(index, elem)
    {
        $(elem).css("cursor", "pointer");
        $(elem).on("click", function(event) {
            var children = $(this).nextUntil(".active");
            $.each($(children), function(index, child)
            {
                if ($(child).is(':visible')) {
                    $(child).hide();
                }
                else {
                    $(child).show();
                }
            });
        });
    });
}

updateFrameHeight();
```

Data Script

```
import java.lang.reflect.Field;
import java.math.BigDecimal;
import java.text.ParseException;
import java.util.Date;
import java.util.Map.Entry;

import org.apache.lucene.document.Document;
import org.ofbiz.core.entity.GenericValue;

import com.atlassian.jira.component.ComponentAccessor;
import com.atlassian.jira.issue.DocumentIssueImpl;
import com.atlassian.jira.issue.Issue;
import com.atlassian.jira.jql.parser.JqlParseException;
import com.atlassian.jira.jql.parser.JqlQueryParser;
import com.atlassian.jira.user.ApplicationUser;
import com.atlassian.jira.user.UserUtils;
import com.atlassian.jira.util.I18nHelper;
import com.atlassian.jira.util.IOUtil;
import com.atlassian.jira.util.json.JSONArray;
import com.atlassian.jira.util.json.JSONException;
import com.atlassian.jira.util.json.JSONObject;
import com.atlassian.query.Query;

import com.decadis.jira.xchart.api.Chart;
import com.decadis.jira.xchart.api.model.Period;
import com.decadis.jira.xchart.api.util.DateUtils;

import com.decadis.jira.xchart.model.MetaCountGroup;
import com.decadis.jira.xchart.model.MetaCountGroups;
import com.decadis.jira.xchart.utils.JiraDateTimeUtils;
import com.decadis.jira.xchart.utils.WorklogUtil;
import com.decadis.jira.xchart.model.param.DateParam;

def formatValues(MetaCountGroups metaCountGroups, DateUtils dateUtils, Period selectedPeriod, String title)
throws JSONException, ParseException
{
  JSONObject jsonObject = new JSONObject();
  JSONArray sums = new JSONArray();
  JSONArray labels = new JSONArray();
  JSONArray columns = new JSONArray();
  JSONArray groupSum = new JSONArray();
  for ( Entry<String, MetaCountGroup> objectList : metaCountGroups.entrySet() )
  {
    groupSum = new JSONArray();
    JSONObject label = new JSONObject();
    label.put("active", true);
    label.put("fontweight", "normal");
    columns.put(label);
    groupSum.put(objectList.getKey());
    for ( Entry<String, BigDecimal> entry : metaCountGroups.getMetaCountGroup().get(objectList.getKey()).
entrySet() )
    {
      groupSum.put(JiraDateTimeUtils.getTimeFormatedString((entry.getValue()).longValueExact()));
    }
    sums.put(groupSum);
    for ( Entry<String, com.decadis.jira.xchart.api.CountGroup> entry : objectList.getValue().entrySet() )
    {
      groupSum = new JSONArray();
      label = new JSONObject();
      label.put("active", false);
      label.put("fontweight", "normal");

      columns.put(label);
      groupSum.put(entry.getKey());
      for ( Entry<String, BigDecimal> entry2 : entry.getValue().entrySet() )
      {
```

```
        groupSum.put(JiraDateTimeUtils.getTimeFormatedString((entry2.getValue()).longValueExact()));
      }
      sums.put(groupSum);
    }
  }

  groupSum = new JSONArray();
  JSONObject label = new JSONObject();
  label.put("active", true);
  label.put("fontweight", "bold");
  columns.put(label);
  groupSum.put("");
  labels.put(title);
  for ( Entry<String, BigDecimal> entry : metaCountGroups.getCountGroup().entrySet() )
  {
    labels.put(dateUtils.getPeriodName(DateUtils.SimpleDateFormat.parse(entry.getKey()), selectedPeriod));
    groupSum.put(JiraDateTimeUtils.getTimeFormatedString((entry.getValue()).longValueExact()));
  }
  sums.put(groupSum);
  jsonObject.put("sum", JiraDateTimeUtils.getTimeFormatedString(metaCountGroups.getSum().longValueExact()));
  jsonObject.put("sums", sums);
  jsonObject.put("labels", labels);
  jsonObject.put("columns", columns);
  return jsonObject;
}

def clear(String string) {
  if ( string == null || string.trim().length() == 0 )
    return "-";
  else
    return string;
}

//here starts the actual script
def i18n = ComponentAccessor.getJiraAuthenticationContext().getI18nHelper();
def user = ComponentAccessor.getJiraAuthenticationContext().getLoggedInUser();
Period selectedPeriod = Period.fromString(TimePeriod);
def DateParam dp = new DateParam();

Date from = dp.getDate(From);
Date to = dp.getDate(To);

JqlQueryParser jqlQueryParser = ComponentAccessor.getComponent(JqlQueryParser.class);
Query query = null;
try
{
  query = jqlQueryParser.parseQuery(JQL);
} catch (JqlParseException e)
{
  throw new IllegalArgumentException("Bad JQL: " + jql);
}

def countGroups = new MetaCountGroups();
def groupValueExtractor = chartBuilder.getGrouper(Grouping);
Field documentField;
try
{
  documentField = DocumentIssueImpl.class.getDeclaredField("document");
  documentField.setAccessible(true);

  for ( Issue issue : chartBuilder.getFilterUtils().performSearch(query, user) )
  {
    if ( issue.getTimeSpent() != null ) {
      for ( String group : groupValueExtractor.getGroups((Document) documentField.get(issue)) )
      {
        group = groupValueExtractor.getResolvedValue(group, issue);
        group = clear(group);

        for ( GenericValue worklog : WorklogUtil.getGenericWorkloads(issue.getId(), from, to) )
        {
          Date startdate = worklog.getTimestamp("startdate");
```

```
          String periodGroup = dateUtils.getPeriodGroup(startdate, selectedPeriod);
          ApplicationUser worker = UserUtils.getUser(worklog.getString("author"));
          String workerName = worker != null ? worker.getDisplayName() : worklog.getString("author");
          countGroups.addValue(BigDecimal.valueOf(worklog.getLong("timeworked")), group, workerName,
periodGroup);
        }
      }
    }
  }
} catch (Exception e) {
  System.err.println("Could not extract groups."+ e);
}

countGroups.fillMissingValues();

try
{
  JSONObject jsonObject = formatValues(countGroups, dateUtils, selectedPeriod, groupValueExtractor.
getGroupName());
  jsonObject.put("title", i18n.getText("common.concepts.time.spent"));
  jsonObject.put("empty", i18n.getText("common.concepts.no.matches"));
  return jsonObject.toString();
} catch (JSONException | ParseException e) {
  System.err.println("JSON Error " + e);
}

return "{ \"error\" : \"Bad Error\"  }";
```

If you still have questions, feel free to refer to our support team.