



Metrics Monitoring Tool Documentation

Layout, Project Data, Charts and Weekly Report Editor

05.02.2015

Tommi Tuominen - 99710
tuominen.tommi.j@student.uta.fi

Table of Contents

1. Introduction	3
2. Main Layout	3
2.1 Project List (project_list.php)	4
2.2 Project Details (project_details.php, subpage of Project List)	5
2.3 Compare Metrics (project_comparison.php)	5
2.4 Weekly Report (readweekly.php)	6
2.5 Redmine (redmine.php)	6
2.6 Facebook (facebook_forum/initialization.php)	7
3. Retrieving project data	8
3.1 Database queries (database_out.php)	8
4. Charts	10
4.1 Making data for the charts (metrics-makedata-1.0.0.js.)	11
5. Weekly report editor	12
5.1 Parser	12
5.2 Adding and hiding fields	13
5.3 Sending the data	13

1. Introduction

This documentation describes the metrics monitoring tool's layout, project data retrieval, weekly report editor and chart functions. All of the PHP files are located in the project's */main/* folder unless stated otherwise. All of the script files are located in */main/scripts/*.

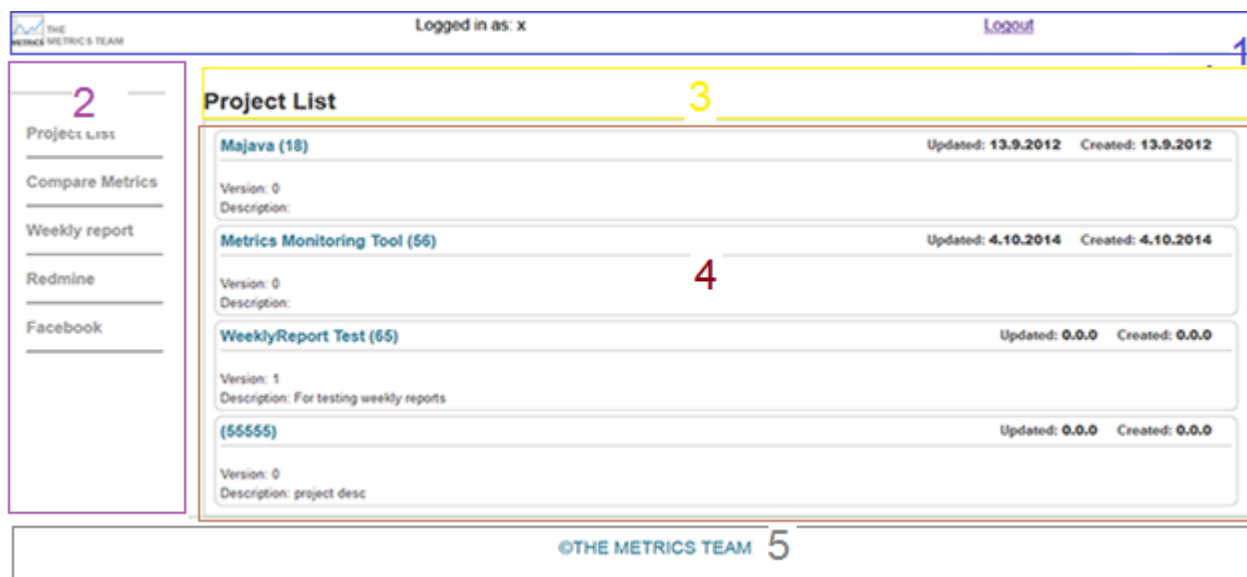
External scripts used in the project:

<http://jquery.com/>

<http://www.highcharts.com/>

2. Main Layout

The layout is designed to be light and easy to use. Basic elements of the website are placed in logical positions for optimal user experience. The layout uses a CSS style file (*main/css/style.css*). Reoccurring elements of the page are created using JavaScript functions to assure easy updates to the page layout.



The layout of the website consists of these main elements:

1. Top
2. Navigation
3. Header
4. Content area
5. Footer

These five main elements are present on every subpage, and are created by using JavaScript functions. Every page uses the script file **metrics-elements-1.0.0.js**, which holds all the functions necessary for creating the elements.

Let's take a closer look at these functions:

```
createTop("<?php Print($user_check); ?>");
```

The **createTop** function creates the top panel of the website. The function takes in one value and uses it as username. Username is defined in the PHP session when logging in. The **createTop** function is also responsible for creating all the other elements in the top panel, such as the logo image and the "sign out" link.

```
createNavig();
```

The **createNavig** function is responsible for creating the website navigation. The creation of navigation links is fully dynamic and adding or deleting links is easy.

```
createHeader("Project List", 1);
```

createHeader creates the header. It takes in two values. The first value is the header text itself and the second value is used for determining if the header has some unique elements. These unique elements can be sorting options or like the fields seen in weekly report page's header.

```
createFooter();
```

Creates the footer section of the website.

Each page has a unique content area, therefore there is no single function populating it. Most of the functions relating the content area are located in **metrics-makedata-1.0.0.js**.

2.1 Project List ([project_list.php](#))

Acts as the main page of the website after the user logs in. This page displays a list of projects and some useful information related to them. The list of projects comes from the metrics database, which is queried to load only the necessary information. Ajax is used for communicating with the php.

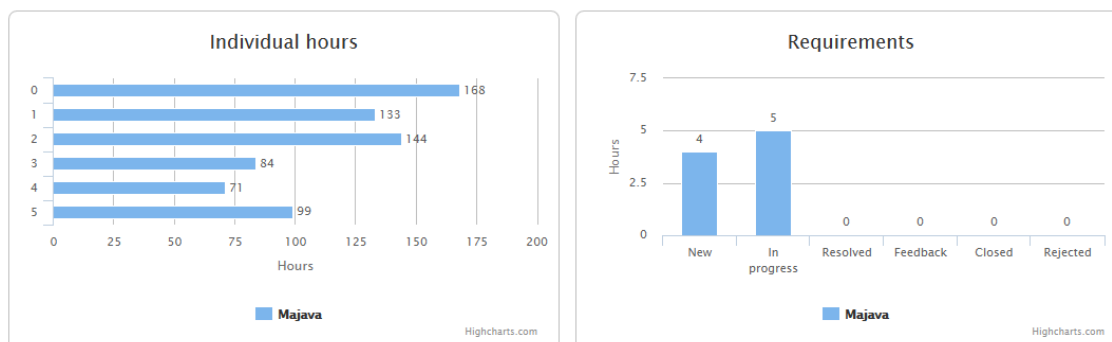
Initiated using this function located in **metrics-makedata-1.0.0.js**

```
getProjectList(0);
```

By clicking a project's name, the user can see information related to that specific project. Project details are shown on the project details page.

2.2 Project Details (project_details.php, subpage of Project List)

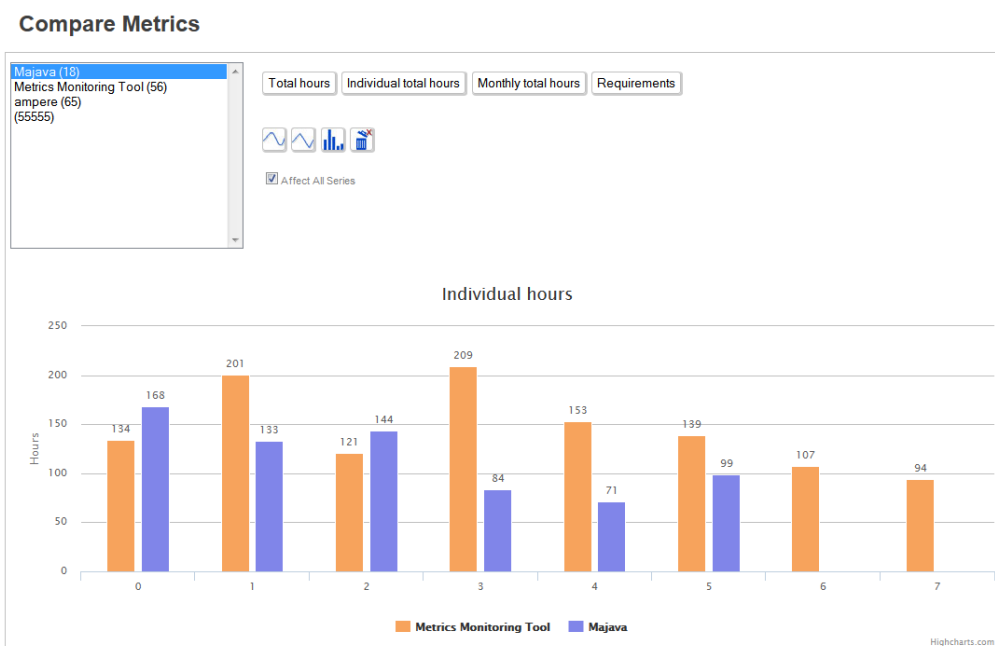
Gets project id from the clicked project and queries the metrics database for project data. Some project details are displayed as charts. The charts are first created as blank dummy charts, and populated after getting actual data from the database.



This page also shows information about project test cases, requirements, code revisions and commits to version control.

2.3 Compare Metrics (project_comparison.php)

The comparison page allows the user to compare different project metrics. Just like with the project list, a list get populated with basic project information. The user can then choose a project from the list and use the given buttons to show wanted metrics. Chart type can be changed between column, line and spline. Chart deletion is also made possible and by unticking the “Affect All Series” the user can affect only the latest series, or a series that has been selected by clicking.



The functions behind all of the buttons are located in **metrics-makedata-1.0.0.js**. When a chart data type button is pressed the code checks for a selected project and fetches the data from the database.

The function takes in an integer that represents the different nature of the button's intended purpose.

`GetSelectedOptions(3)`

The chart type buttons use a function that updates the current chart with the new selected type.

2.4 Weekly Report (readweekly.php)


The weekly report editor is built to parse information from a text file and show it in a more comprehensive and easily editable format. Parsing from a text file is optional, since creating a completely new weekly report is also possible.

Creating a new weekly report is fairly simple, the user can add fields and textboxes that fit their predefined roles. The functions used on this site are located in **read_weeklyreport.js**

After finishing the weekly report, the user can upload it to the database.

2.5 Redmine (redmine.php)

This page is used for manually updating the metrics database with data from redmine. A progress bar is displayed in the top right corner of the site to indicate the process.

Processing individual working hours...

 494 row(s) processed.

Project List

Compare Metrics

Weekly report

Redmine

Facebook

Redmine

Individual_work :

already imported in local database : 934
 numbers of data in redmine database : 933

Requirements :

already imported in local database : 34
 numbers of data in redmine database : 30

Projects :

already imported in local database : 4
 numbers of data in redmine database : 2

The redmine page also shows the current amount of data in both redmine and metrics database.

2.6 Facebook (/facebook_forum/initialization.php)

The Facebook page is used for getting information from Facebook groups. The user can limit fetching dates and choose a certain person in a group. Facebook group id is needed for fetching the data.

Facebook Query

Date from:

2014-01-01

Date until:

2015-02-04

Group ID:

652060828245934

Everyone ▼

Update member list in DB

Add group to list

Recent posts:

5



Lähetä kysely

3. Retrieving project data

All interaction with the database is handled with JQuery's Ajax functions. There are two Ajax functions: the other one is for getting a list of projects and the other one is for more customisable queries. Both of the functions use Post method for passing the variables to **database_out.php**. These two Ajax functions are located in **metrics-makedata-1.0.0.js** and they are called:

getProjectList(options)

and

getProjectData(id,operation,querytype,value,containername)

getProjectList takes in an integer variable that defines what to do with the information after it is successfully retrieved. If the variable is set to 1, it means that the data is for the comparison page and if the variable is 0, the data is for the main page's project list.

getProjectData takes in project id, operation to be executed in php, query type for handling database query, value; and container name for chart creation after successful retrieval of data.

3.1 Database queries (database_out.php)

The database_out.php file is used for fetching data from the metrics database. The php file includes functions for selecting from *weekly_report*, *weekly_report_requirement*, *project*, *participation*, *member*, *requirement* and *individual_work*. It gets project_id, querytype and operation variables by Post method. After checking the database connection and deeming it successful, the code checks the wanted query type.

echoResults(\$operation, \$project_id, \$con, \$scope, \$where, \$equal)

If the variable querytype is 0, the code runs a function called **echoResults**, which is used for querying only single tables. The **echoResults** function takes in an integer variable "operation", that defines which table will get echoed. The other values are: project id, database connection, scope; where and equal (used for select query).

getDataForCharts(\$project_id, \$con)

Else if the querytype is 1, **getDataForCharts** function is called. the **getDataForCharts** function is used for getting information from multiple tables and packing them all into an object.


```

getWeeklyReports($project_id, $con, $scope, $where, $equal)
getWeeklyRequirements($project_id, $con, $scope, $where, $equal)
getProjects($project_id, $con, $scope, $where, $equal)
getParticipation($project_id, $con, $scope, $where, $equal)
getRequirements($project_id, $con, $scope, $where, $equal)
getIndividual($project_id, $con, $scope, $where, $equal)

```

The code consists of functions that all have their own respective table to query. The functions used for table queries all take in the same values. All of the functions use normal MySQL selecting queries, and all the variables they take in are related to controlling those queries. All the information the code gets from a table is put into an array. The arrays containing the data are then encoded into a Json object and sent back to the Ajax function.

Sort by key

```

+ weekly_report

- project
  - 0
    project_id
    project_name
    created_on
    updated_on
    status
    version
    description
  participation
+ individual

+ requirement

weekly_requirement

```

The project objects are constructed as shown above. Each table gets its own key and sub keys for all information in that table regarding selected project id.

makeLineData(value, containername)

After successfully getting all project data, **makeLineData** function is called. It takes in two values. First value is an integer and it determines what kind of chart needs to be drawn and the second value is the name of the div container housing the chart.

4. Charts

The charts are made possible by an external JavaScript library called **HighCharts**.

The HighCharts library has many built-in functions for manipulating the charts. These built-in functions include updating the chart's data and type without completely re-drawing the whole chart. However, custom functions for controlling the built-in functions had to be created for easier presentation and modification of the charts. All of the custom code regarding the charts can be found in **metrics-makechart-1.0.0.js**.

CreateChart(type, x_label, y_label, container_id, chart_title)

The **CreateChart** function is used for creating a chart. It takes in five values. First value dictates the type of the chart. Type is given as string ("bar", "column", "line", "spline", ...). 2nd and 3rd values are used for labelling the x and y axis. container_id is the string id of the html div container to house the chart object. chart_title is the string title of chart. The chart created with this function is merely a dummy chart waiting for real data and values.

addLine(DataArray, xCategories, containername, chart_title, types, seriesname, chartno)

The **addLine** function is used for adding a new series to a chart. The function takes in DataArray, which contains the data that will be shown on the chart. xCategories is an array used for naming the x axis categories. containername is the string name of the container where the chart is. chart_title string defines the title of the chart. types is an integer that is used for detecting if the chart needs to be redrawn. seriesname string defines individual series names. chartno integer is used for identifying the right chart if there are multiple charts on one page.

removeSeries()

removeSeries function is used for removing all series from a chart. Optionally, if "affect all" is not checked, only the selected series will be deleted. By default the latest series is always selected.

change(newstyle)

change function is used for changing the chart's visual style. It takes in a string value, which is directly used as the new type. The new style can be any of the styles described in highcharts documentation. The styles used in this project are bar, column, line and spline.

There is also a function for creating a test chart with random data. This function is called:

ChartWithRandomData(type)

It takes in a type string, which dictates the type of the chart just like in the CreateChart function. Data is randomized using a function called:

RandomizeData()

4.1 Making data for the charts (metrics-makedata-1.0.0.js.)

Data from the database is not initially ready to be displayed by the charts. The HighCharts charts can read objects, but the objects must have predefined key names in order for them to work properly. The data also needs to be calculated and right x-categories need to be chosen. All this is made in **metrics-makedata-1.0.0.js**.

There are 8 main functions that all have their own role in making the data into the right, displayable format. These functions are as follows:

```
getMonthlyHours()
getIndiHours()
getAllHours()
getRequirements()

projectRequirements()
projectTestcases()
projectCoderevisions()
commitsToVersionCtrl()
```

They all have their own role and calculations, but they all use the same project object that has been retrieved from the database. These functions are called in the **makeLineData(value, containername)** function, in the makeLineData function there is first a check to see what kind of data needs to be drawn. For example if there is a chart for monthly hours, then only the **getMonthlyHours()** function will be run. The object format for all the chart data is: {y: int, name: string}. The y key is the value of the line point and the name is the name of that point.

Other functions in the **metrics-makedata-1.0.0.js** file are:

```
parseDate(date)
parseDateTime(scope, datetime)
```

These functions are used for parsing and making sense of the date values in the database. The date values in the database are in a rather unusable format. **parseDate(date)** takes in the raw date format and uses the **parseDateTime** function to make it into dd.mm.yyyy format.

```
clearArrays()
```

Clears arrays.

```
createOptions(name,id)
getSelectedOptions(value)
```

createOptions(name,id) creates the list of projects in the comparison page. Name and id come from the queried project list object.

getSelectedOptions(value) checks which project in the list is selected and calls the **getData** function, which houses the Ajax function used to retrieve project information.

5. Weekly report editor

The weekly report editor uses `readweekly.php` file and its functions are located in `read_weeklyreport.js`. One of the main functionalities of this editor is the ability to read and parse text files. After parsing a text file the page shows all the parsed information in a more readable and editable form. The editor also allows creation of new text fields, which means that parsing information from a text file is optional.

5.1 Parser

readSingleFile(evt)

Is the parser function. The parser searches the text file for keywords indicated with #. When a keyword is found, all data after it are considered nodes belonging to that keyword. Many of the keyword categories have different kinds of functions related to them, like for example, some require textboxes and some need two text fields instead of one.

getData(i, regex, idprefix, arrayOfLines, detectedkey)

getData is called every time the parser hits a keyword. It is responsible for looping through sub lines of the detected key word. It takes in 5 values:

`i` (int) nth round of the parser loop.

`regex` (string) determines which regular expression to use. mainly uses email regex.

`idprefix` (string) prefix for unique id.

`arrayOfLines` (array) all the text file lines.

`detectedkey` (string) detected keyword.

It then creates the needed fields into their own respective container divs using the **CreateInput** function.

CreateInput(text,idprefix,order,counter,detectedkey,classname)

CreateInput function is responsible for creating input fields on to the page. It takes in 6 values:

`text` (string) used as input value attribute.

`idprefix` (string) used as a prefix for unique element ids .

`order` (int) used to detect if a line break is needed.

`detectedkey` (string) detected key word.

`classname` (string) used for giving non-unique class names to fields.

document.createElement("input");

document.createElement("textarea");

Fields are then created using JavaScript's own **createElement** function.

5.2 Adding and hiding fields

addToSection(detectedkey)

This function is used when adding new input fields into the report. It takes in the detected key which is used to identify what kind of field the user wants to add. The **addToSection** function then calls for **CreateInput** with the right values.

hideSection(targetid, headerid, btnid)

Hides the wanted section of weekly report. Does not really serve any purpose other than visual. Takes in values required for identifying the right target div, header and button.

5.3 Sending the data

clicked()

When the user clicks the submit button **clicked()** is called. This function reads all the text fields and constructs an object that will be sent to the server using Ajax.

JSON

```
+ client
+ completed_tasks
+ managers
- milestone
  + 0
  - 1
    name
+ other_test
+ otherinfo
+ problems
+ requirements
+ revisions
+ tasks_next
+ unit_test
+ workinghours
```

Above is an example object created from the data in the weekly report editor.

Documentation about the keywords and sending the data into the metrics database will be given by Jueran Huang.