Sergii Krymets

# Rainfall Volume Calculator Web Application Design

v1.0

Sergey
2018-06-12

## Contents

## Prerequisites

Before attempting this you should have installed all the necessary software.

Make sure that you have installed and configured these tools, which will be necessary for developing the Calculator Application and following along with the rest of this document:

- JDK 8
- Maven 3.5.*
- Unpacked distribution of Wildfly 10 Application Server
- Git

Please follow the corresponding manuals for that software packages to properly setup and configure it.

## Getting Started

The first thing we need to do is get a copy of the project. If you have git installed on your computer, to clone the project run:

```
$ git clone https://github.com/skrymets/crx.git
```

This will create project folder on you file system.

### Project Folder Structure

Inside the copied project folder you should find the next content

```
$> cd crx
$> ls
crx-calc/  crx-web/  doc/  pom.xml
```

| | |
|---|---|
| **crx-calc/** | This folder contains sources of library that implements the calculator algorithm. |
| **crx-web/** | This folder contains sources of Enterprise Web Application project, that exposes the algorithm to the WEB, by providing convenient REST API and UI for the and users. |
| **doc/** | This folder contains documentation for whole project. |
| **pom.xml** | Maven build script. |

## Web Application Build Steps

There are a few contexts in which you may will build the Calculator Web Application (crx-web) project: **UI Development**, **Backend Development and Testing**, **Final Build**.

Regarding to a particular context you would or wouldn't to activate some particular profiles of project's POM.

| Profile Name | Intention | Active by Default |
|---|---|---|
| default | General purpose profile | Yes |
| application-server | This profile enables application do be deployed into an instance of Application Server, make debugging, UI and Backend development with hot redeploy and so on. | Yes |
| application-server-dist | This profile downloads and unpacks an instance of Application Server, required to run " application-server" | No |

|  |  |  |
|---|---|---|
| ejb-test | profile. It activates automatically in the case if a distribution of application server does not exist in a /dist folder. In the case when the distribution already exists there - the profile remains inactive.<br>This profile is required to run Arquillian tests for business services and/or REST endpoints. Activate it when you add new business logic and API endpoints with the corresponding tests. | No |

| Context | Build Steps |
|---|---|
| **Final Build** | mvn clean install |
| **Backend Development and Testing** | mvn clean test -Pejb-test |
| **UI Development** | mvn clean install<br>mvn gwt:run<br><br>or<br><br>mvn clean install<br>mvn gwt:debug |

## Source Code Organization

### crx-calc

This is a simple maven project which folder structure fully corresponds Maven's standard project structure for packaging "jar".

### crx-web

This is a **war archetype** maven project which folder structure fully corresponds Maven's standard project structure for packaging "war".

| Source Package | Content description |
|---|---|
| com.crxmarkets.api | Contains JAX-RS related application classes, like: JAX-RS activator, and Calculator REST Resource Implementation. |
| com.crxmarkets.dao | EJB that provides persistence in the application |
| com.crxmarkets.services | EJB that provides application's main business logic |
| com.crxmarkets.web | Contains ERRAI/GWT module descriptor |
| com.crxmarkets.web.client.local | User Interface classes and templates which are being translated by ERRAI into the client-side HTMLs and JavaScript files that constitute the Rich Client UI.<br>Everything that should be compiled to JavaScript to be available on the client side, should reside under this package.<br>Content of those packages is not deployed on the server side. |
| com.crxmarkets.web.client.shared | Service interfaces (like JAX-RS) persistence entities and so on, that is shared between client and server side. ERRAI will build the required proxies for all this stuff and make it available both: on server side and client side. |

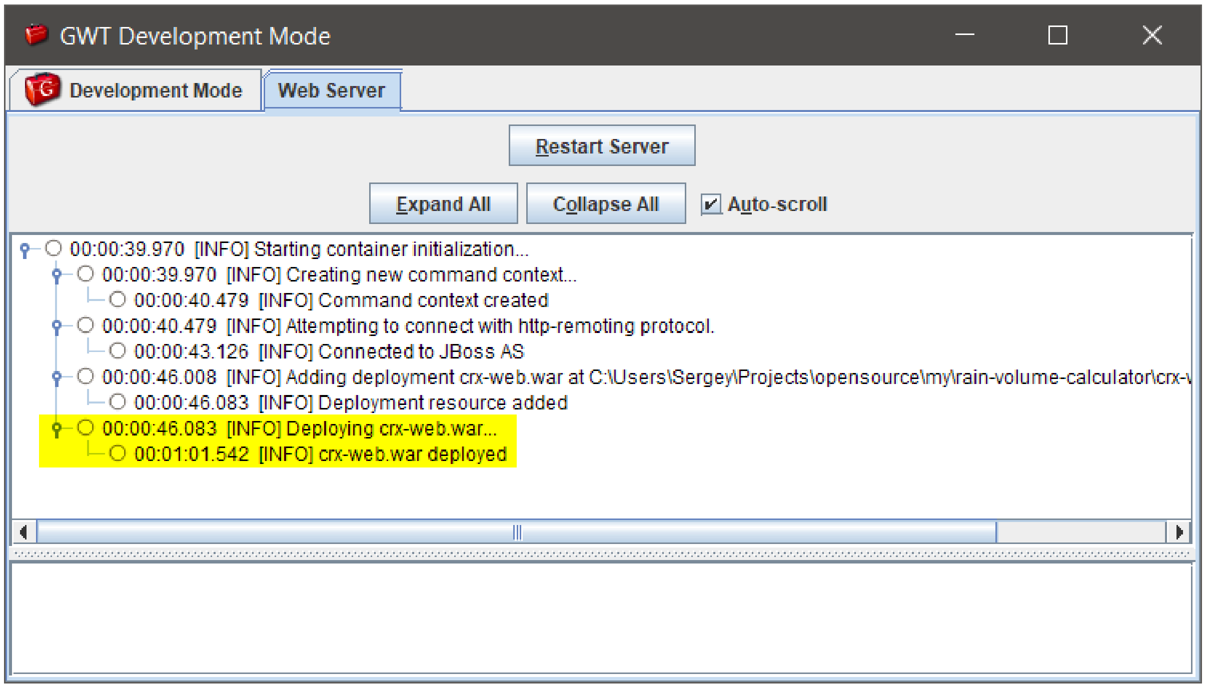
| | |
|---|---|
| com.crxmarkets.arquillian | Arquillian tests for backend EJBs, services etc. This package is excluded from the build until you activate **ejb-test** maven profile. |

## Start the development

Build the application and run it in the GWT "DevMode". Type in console

```
$> mvn clean install
$> mvn gwt:run
```

This will run GWT Code Server along with Application Server with the web application deployed on it.



```
00:00:39.970  [INFO] Starting container initialization...
  00:00:39.970  [INFO] Creating new command context...
    00:00:40.479  [INFO] Command context created
  00:00:40.479  [INFO] Attempting to connect with http-remoting protocol.
    00:00:43.126  [INFO] Connected to JBoss AS
  00:00:46.008  [INFO] Adding deployment crx-web.war at C:\Users\Sergey\Projects\opensource\my\rain-volume-calculator\crx-\
    00:00:46.083  [INFO] Deployment resource added
  00:00:46.083  [INFO] Deploying crx-web.war...
    00:01:01.542  [INFO] crx-web.war deployed
```

Open your browser and follow this address:

http://localhost:8888/crx-web/

## Building and Deploying as a Standalone Application

### Starting WildFly

To run this application, it should be deployed to the WildFly Application Server. To start a standalone instance of WildFly, run:

```
$> $JBOSS_HOME/bin/standalone.sh
```

where $JBOSS_HOME is the path to WildFly folder you unzipped its distribution.

## Build

Switch to the project folder, and run the command:
Whait until you see this in the output:

```
[INFO] ------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------
```

Now it's time to deploy the application. While the WildFly server instance is running, run the command:

```
$> cp crx-web/target/crx-web.war $JBOSS_HOME/standalone/deployments/
```

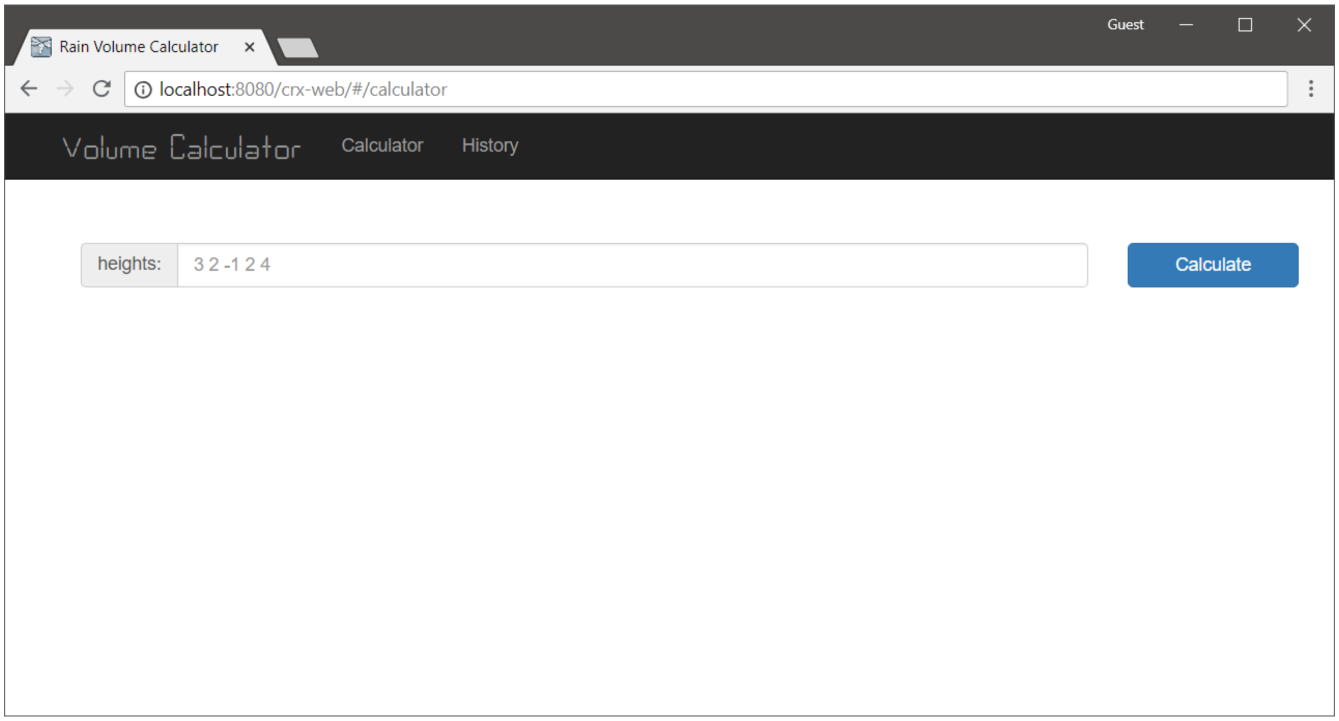
If in a while in the server logs you can observe output like this:

```
INFO  [org.wildfly.extension.undertow] (ServerService Thread Pool -- 70) WFLYUT0021: Registered web context: /crx-web
INFO  [org.jboss.as.server] (DeploymentScanner-threads - 2) WFLYSRV0010: Deployed "crx-web.war" (runtime-name : "crx-web.war")
```

this means that Calculator Application was deployed successfully.
Open your browser and follow this address:

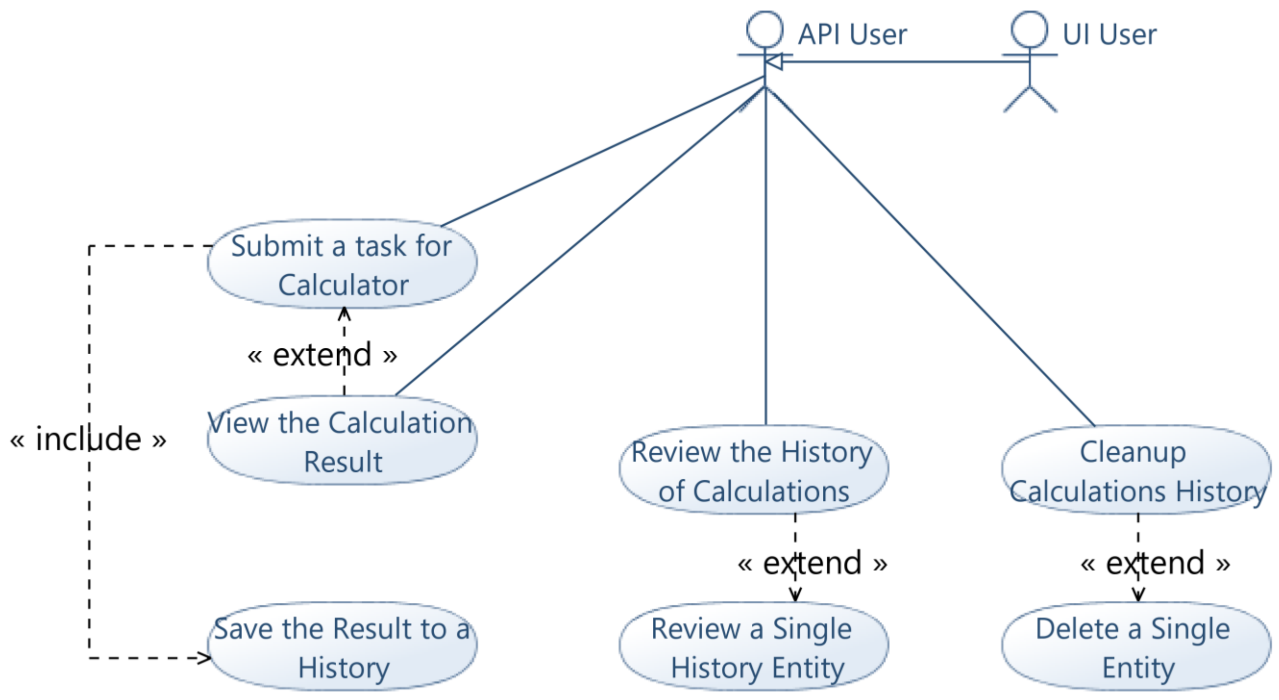
http://localhost:8080/crx-web/

You will see the default page of the application:



## Application Use Cases

The overall scope of Use Cases provided by this application is depicted on this diagram.

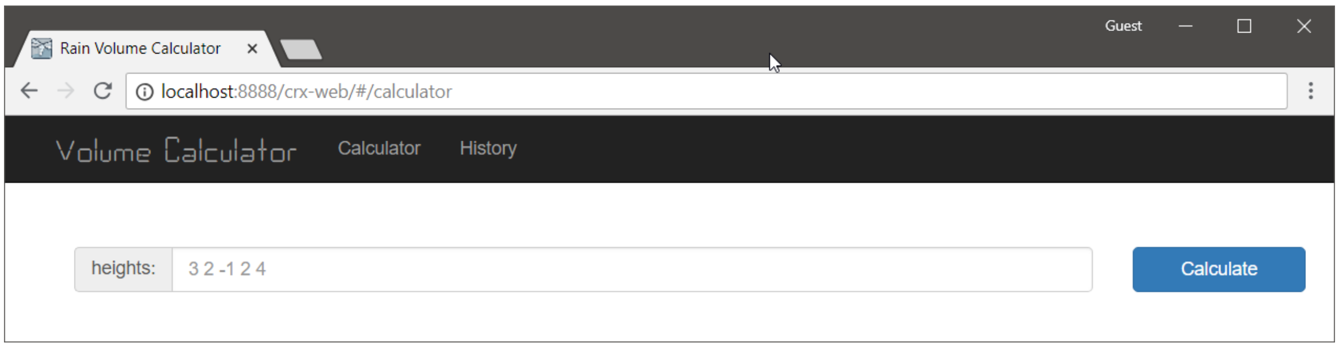

API User
UI User
Submit a task for Calculator
« extend »
View the Calculation Result
« include »
Review the History of Calculations
Cleanup Calculations History
« extend »
« extend »
Save the Result to a History
Review a Single History Entity
Delete a Single Entity

### Participants

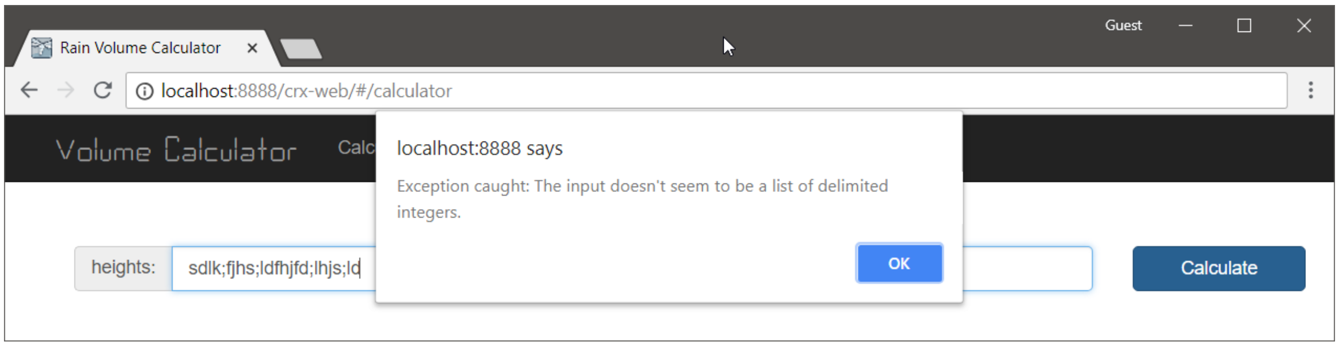
There are two kind of participants in the application: **UI User** and **REST API User**. UI User can perform only those operations which are exposed through the REST API.

### Case: Submit a task for Calculator

In case to perform a calculation, UI user starts from the main page of the application.



The user should input a task, which is a list of integers that represent a two-dimensional surface. The application will try it's best to recognize the input: UI User can use a various delimiter characters like: « », «,» «|», «;» or altogether to delimit numbers in a list. «1 3 4 -8  5 7 2 0 -3 9 4 5 7 2 0 9  3» will work, as well as «4, 3, 9, 0, 8, -6, 7, 3, 4, 6, -3, 0, 9, 7», and even «2 0; 3, 4; 9 -6| 7| 2, 0; 9| 7, 3; 0,    9». In all other cases application will rise an error.



localhost:8888 says

Exception caught: The input doesn't seem to be a list of delimited integers.

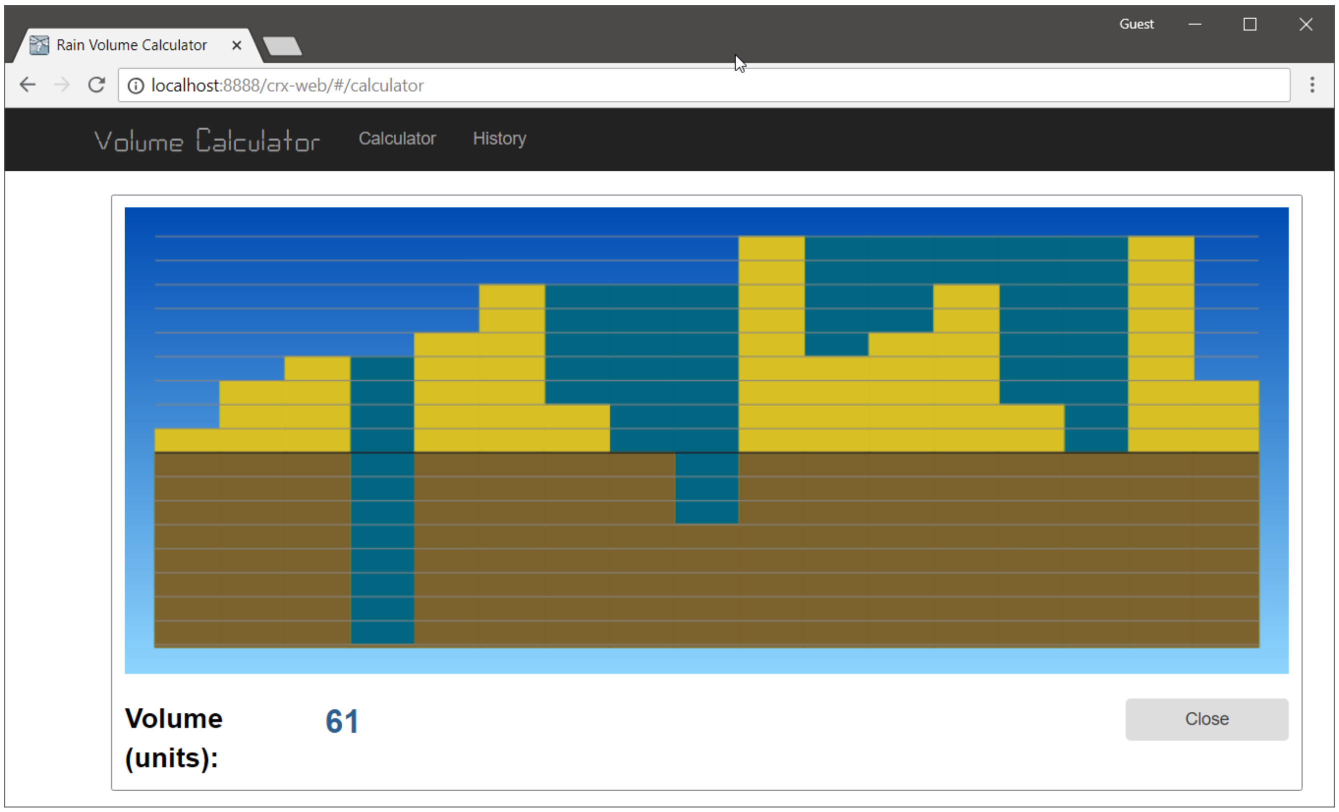
## REST Endpoint

**POST**

- **Path:** calculator/task
- **Payload Class**
    - com.crxmarkets.web.client.shared.CalculationTask
- **Headers**
    - Content-Type: application/json
- **Body**
    - {
      "heights":[-1,-2,-1,-2,-1,-2,10]
      }
- **Response**
    - com.crxmarkets.web.client.shared.CalculationResult
    - 200 OK
        - {
          "totalVolume":3,
          "input":[-1,-2,-1,-2,-1,-2,10],
          "levels":[-1,-1,-1,-1,-1,-1,10]
          }
    - 400 Bad Request
        - Task data is incomplete

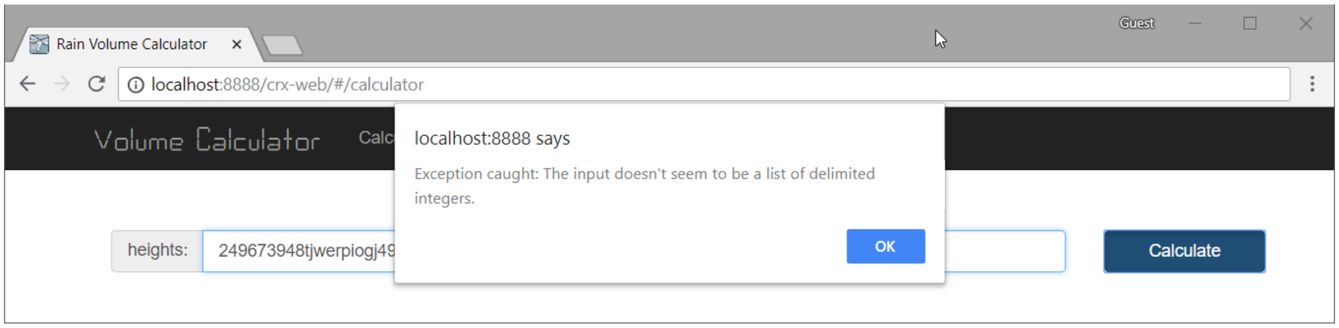
## Case: View the Calculation Result

If user provides the correct input - he will see the histogram, the represents the calculation result: a total number of rainfall in units, and the corresponding lakes visualization.



**Volume (units):** 61

IN the case of incorrect input, he will see a notification about this:



localhost:8888 says

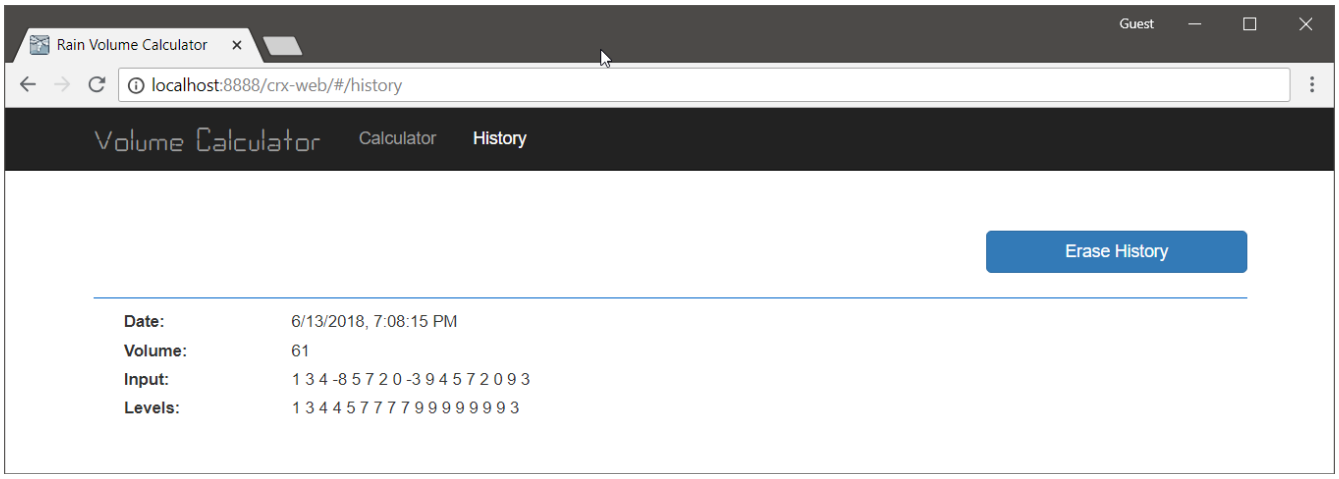
Exception caught: The input doesn't seem to be a list of delimited integers.

### REST Endpoint

See **REST Endpoint** section of " Case: Submit a task for Calculator".

## Case: Save the Result to a History

Any calculation that was executed correctly from the UI client is automatically saved in the history.



| | |
|---|---|
| Date: | 6/13/2018, 7:08:15 PM |
| Volume: | 61 |
| Input: | 1 3 4 -8 5 7 2 0 -3 9 4 5 7 2 0 9 3 |
| Levels: | 1 3 4 4 5 7 7 7 7 9 9 9 9 9 9 9 3 |

## REST Endpoint

**POST**

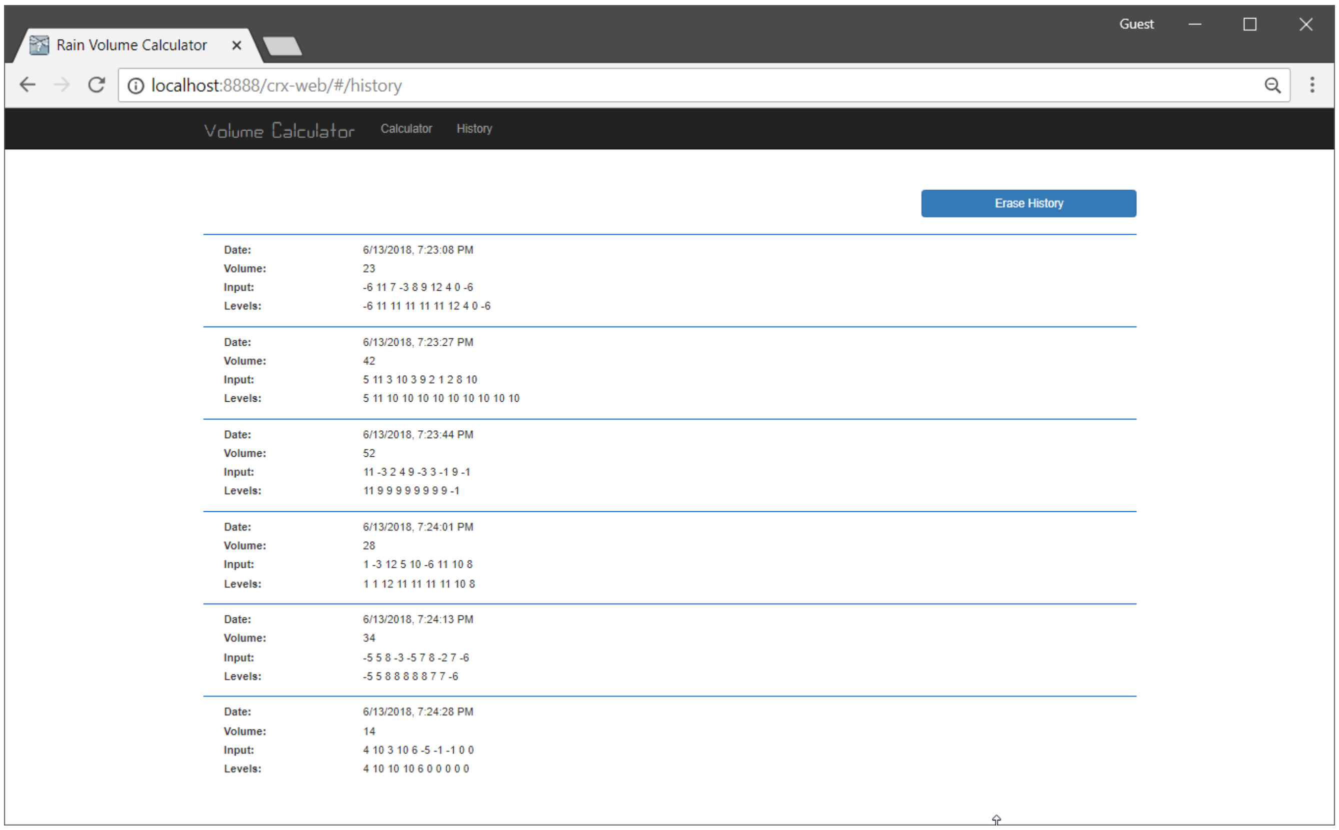
- **Path:** /calculator/history
- **Payload Class**
  - com.crxmarkets.web.client.shared.HistoryItem
- **Headers**
  - Content-Type: application/json
- **Body**

  -
    ```
    {
    "task":"2 3 9 0  8 4 5 7 2 3 4 5",
    "calculation": "2 3 9 8 8 7 7 7 5 5 5 5",
    "total":19,
    "dateTime":1528869910808
    }
    ```
- **Response**
  - 200 OK
    - An integer number, which represents the ID of newly created history item
  - 400 Bad Request
    - History item data is incomplete.

## Case: Review the History of Calculations

To review the history of calculations an UI user should follow this link:

http://<hostname>:<portname>/crx-web/#/history



| | |
|---|---|
| **Date:** | 6/13/2018, 7:23:08 PM |
| **Volume:** | 23 |
| **Input:** | -6 11 7 -3 8 9 12 4 0 -6 |
| **Levels:** | -6 11 11 11 11 11 12 4 0 -6 |

| | |
|---|---|
| **Date:** | 6/13/2018, 7:23:27 PM |
| **Volume:** | 42 |
| **Input:** | 5 11 3 10 3 9 2 1 2 8 10 |
| **Levels:** | 5 11 10 10 10 10 10 10 10 10 10 |

| | |
|---|---|
| **Date:** | 6/13/2018, 7:23:44 PM |
| **Volume:** | 52 |
| **Input:** | 11 -3 2 4 9 -3 3 -1 9 -1 |
| **Levels:** | 11 9 9 9 9 9 9 9 9 -1 |

| | |
|---|---|
| **Date:** | 6/13/2018, 7:24:01 PM |
| **Volume:** | 28 |
| **Input:** | 1 -3 12 5 10 -6 11 10 8 |
| **Levels:** | 1 1 12 11 11 11 11 10 8 |

| | |
|---|---|
| **Date:** | 6/13/2018, 7:24:13 PM |
| **Volume:** | 34 |
| **Input:** | -5 5 8 -3 -5 7 8 -2 7 -6 |
| **Levels:** | -5 5 8 8 8 8 8 7 7 -6 |

| | |
|---|---|
| **Date:** | 6/13/2018, 7:24:28 PM |
| **Volume:** | 14 |
| **Input:** | 4 10 3 10 6 -5 -1 -1 0 0 |
| **Levels:** | 4 10 10 10 6 0 0 0 0 0 |

## REST Endpoint

**GET**

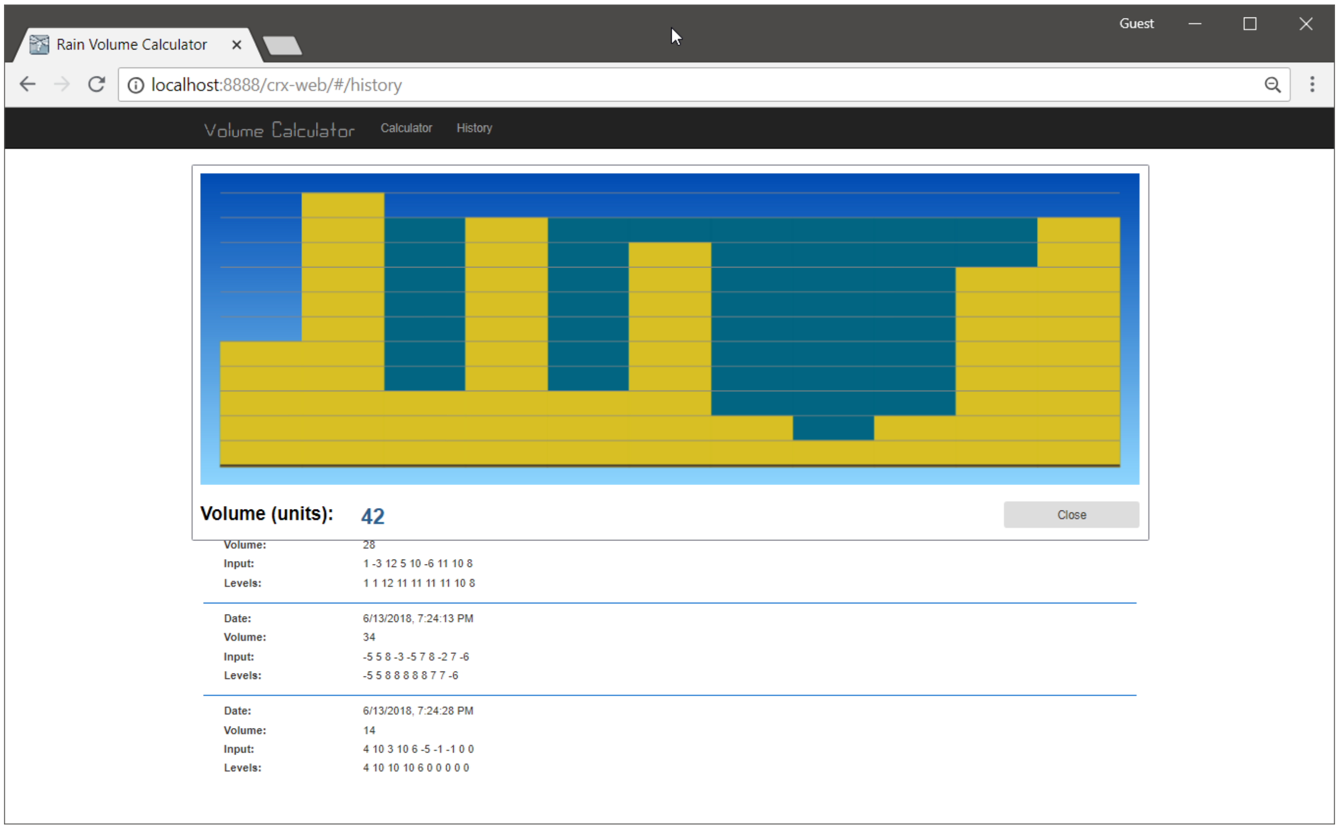
- **Path :** /calculator/history
- **Response**
  - **200 OK**
    - **List<com.crxmarkets.web.client.shared.HistoryItem>**

      -
        ```
        [
          {
            "id": 1,
            "task": "2 3 9 0  8 4 5 7 2 3 4 5",
            "calculation": "2 3 9 8 8 7 7 7 5 5 5 5",
            "total": 19,
            "dateTime": null
          },
          {
            "id": 2,
            "task": "2 3 9 0  8 4 5 7 2 3 4 5",
            "calculation": "2 3 9 8 8 7 7 7 5 5 5 5",
            "total": 19,
            "dateTime": null
          },
          {
            "id": 3,
            "task": "3 6 7 3 0 5 8",
            "calculation": "3 6 7 7 7 7 8",
        ```

```
        "total": 13,
        "dateTime": 1528869893049
    },
    {
        "id": 4,
        "task": "0 4 9 6 7 4",
        "calculation": "0 4 9 7 7 4",
        "total": 1,
        "dateTime": 1528869910808
    }
]
```

## Case: Review a Single History Entity

If an UI User clicks on a single history item in the History List page - he can review calculation results visualization.



Volume (units): 42

| | |
|---|---|
| Volume: | 28 |
| Input: | 1 -3 12 5 10 -6 11 10 8 |
| Levels: | 1 1 12 11 11 11 11 10 8 |
| Date: | 6/13/2018, 7:24:13 PM |
| Volume: | 34 |
| Input: | -5 5 8 -3 -5 7 8 -2 7 -6 |
| Levels: | -5 5 8 8 8 8 8 7 7 -6 |
| Date: | 6/13/2018, 7:24:28 PM |
| Volume: | 14 |
| Input: | 4 10 3 10 6 -5 -1 -1 0 0 |
| Levels: | 4 10 10 10 6 0 0 0 0 0 |

### REST Endpoint

**GET**

- **Path:** /calculator/history/{id}
- **200 OK**

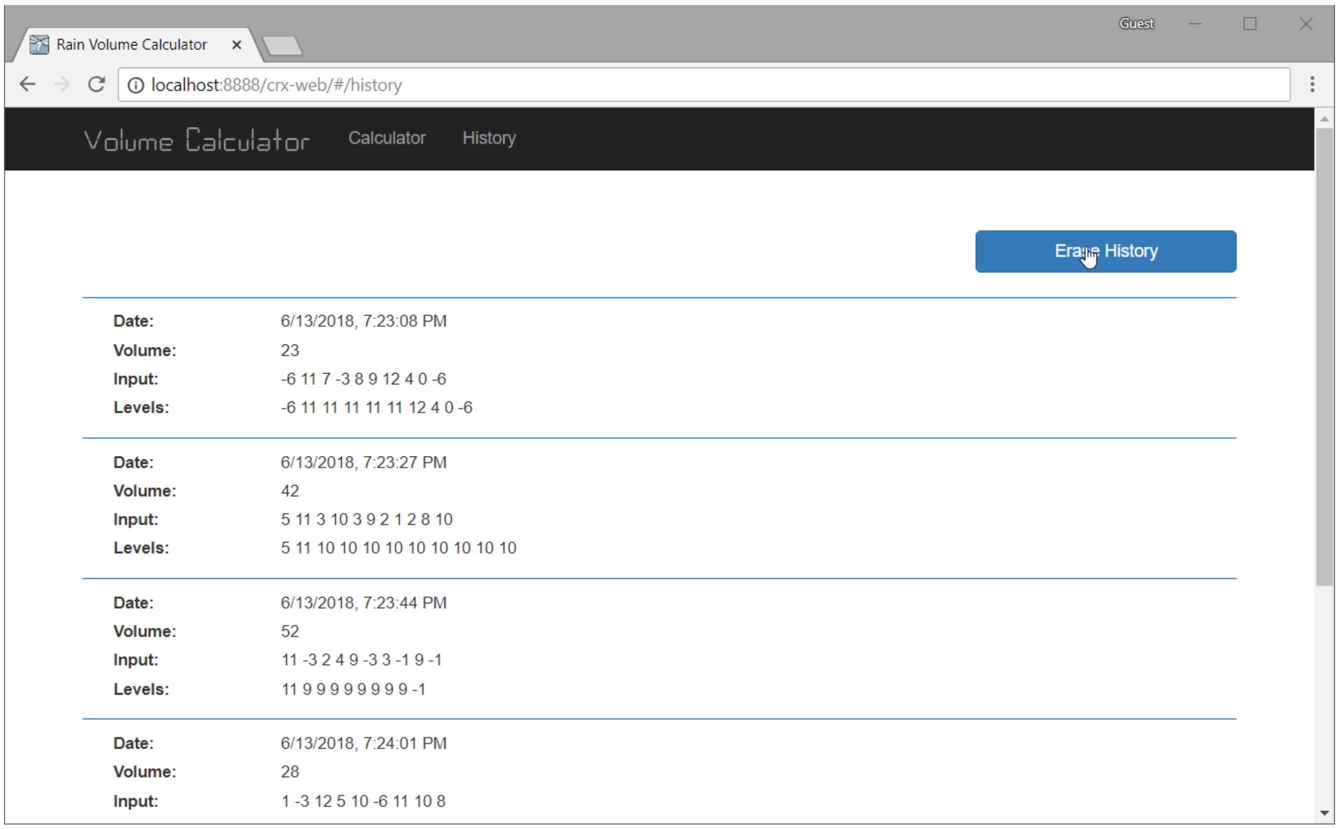
  -
    ```
    {
        "id": 1,
        "task": "2 3 9 0  8 4 5 7 2 3 4 5",
        "calculation": "2 3 9 8 8 7 7 7 5 5 5 5",
    ```

```
        "total": 19,
        "dateTime": 110109284021
      }
```

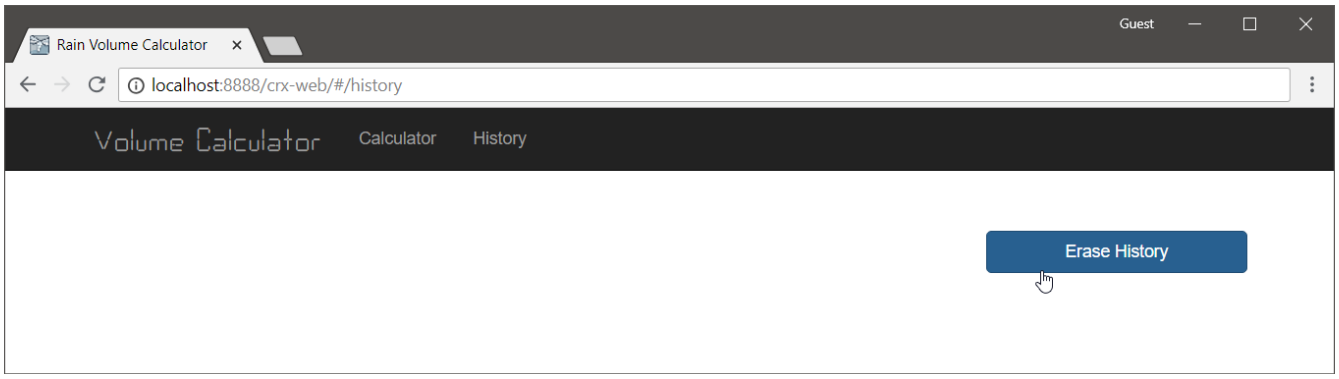
- **404 Not found**
    - In case if the provided ID does not reference any History Item

## Case: Cleanup Calculations History

By clicking to "Erase History" button UI user can cleanup all items saved in the History List.



| | |
|---|---|
| Date: | 6/13/2018, 7:23:08 PM |
| Volume: | 23 |
| Input: | -6 11 7 -3 8 9 12 4 0 -6 |
| Levels: | -6 11 11 11 11 11 12 4 0 -6 |
| Date: | 6/13/2018, 7:23:27 PM |
| Volume: | 42 |
| Input: | 5 11 3 10 3 9 2 1 2 8 10 |
| Levels: | 5 11 10 10 10 10 10 10 10 10 10 |
| Date: | 6/13/2018, 7:23:44 PM |
| Volume: | 52 |
| Input: | 11 -3 2 4 9 -3 3 -1 9 -1 |
| Levels: | 11 9 9 9 9 9 9 9 9 -1 |
| Date: | 6/13/2018, 7:24:01 PM |
| Volume: | 28 |
| Input: | 1 -3 12 5 10 -6 11 10 8 |



### REST Endpoint

**DELETE**

- **Path:** /calculator/history
- **200 OK**

## Case: Delete a Single Entity

This functionality is available only via REST Interface.

### REST Endpoint

**DELETE**

- **Path**: /calculator/history/{id}
- **200 OK**
- **404 Not Found**
    - In case when the History item can not be found