Crowdsourcing framework
Json API documentation

August 10, 2015

Chapter 1

GET interface

This section will explain how to use the get json interface. The example figures will be constitued by the input code on the left, and the output given by the platform on the right.

1.1 Access the interface

To access the interface, you will have to send a POST request to :

<framework root>/json/?page=get

The POST data must content only json, and the request *Content-type* header must be set to *application:json*.

1.2 login

All json requests must contain a *login* object to authenticate yourself as an administrator of the platform.

```
{
  "login":{
    "id":"admin",
    "password":"admin"
  }
}
```

This input alone will return an error, as the script expects a request. It is, however, required to include it in any request.

Figure 1.1: Example of login code

1.3 Make a request

The request is made in a *request* object that contains at least a *type* field.

```
{
  "login":{
    "id":"admin",
    "password":"admin"
  },
  "request":{
    "type":"worker"
  }
}
```

```
[
    {
        "id": "1",
        "username": "worker",
        "password": "$2y$10$IuB4WsobsGdGNCo1ia",
        "birthdate": "1993-07-04"
    },
    {
        "id": "2",
        "username": "worker2",
        "password": "$2y$10$p0U3pW52oWaFN4qESD",
        "birthdate": "2000-05-04"
    }
]
```

Figure 1.2: Example minimal request

As shown on the example above, the return is an array of the matching data.

1.4 Specifying return fields

If you don't need all the return fields, you can use the following syntax to specify the ones you want :

```
{
  "login":{
    "id":"admin",
    "password":"admin"
  },
  "request":{
    "type":"worker",
    "fields":[
      "id",
      "username"
    ]
  }
}
```

```
[
    {
        "id": "1",
        "username": "worker"
    },
    {
        "id": "2",
        "username": "worker2"
    },
    {
        "id": "3",
        "username": "worker3"
    }
]
```

Figure 1.3: Example request with return fields selection

1.5 Filtering the results

The results can be filtered using the following syntax :

```
{
  "login":{
    "id":"admin",
    "password":"admin"
  },
  "request":{
    "type":"worker",
    "fields":[
      "id",
      "birthdate"
    ],
    "filters":[
      "DATEDIFF(NOW(),birthdate)>(20*365)"
    ]
  }
}
```

```
[
    {
        "id": "1",
        "birthdate": "1993-07-04"
    },
    {
        "id": "3",
        "birthdate": "1960-10-22"
    }
]
```

Figure 1.4: Example request with results filtering

The example above selects only the users whose age is greater than 20. The filters must be written with SQL syntax. There can be as many filters as needed. Fields selection and filters can be used together or separately.

1.6 sub-results

Some types (*task*, *questions* and *contributions*) have sub-results. Fields selection and results filtering can be applied to these sub-results by adding sub-objects to the *fields* or *filters* objects as shown in the following example :

```
{
  "login":{
    "id":"admin",
    "password":"admin"
  },
  "request":{
    "type":"task",
    "fields":{
      "0":"name",
      "questions":{
        "0":"question",
        "answers":"false"
      }
    }
  }
}
```

```
[
    {
        "id": "1",
        "name": "test_task 1",
        "questions": [
            {
                "id": "1",
                "question": "How are you ?"
            },
            {
                "id": "2",
                "question": "is this picture
                    correctly displayed?"
            }
        ]
    },
    {
        "id": "2",
        "name": "assignTest",
        "questions": [
            {
                "id": "3",
                "question": "test"
            }
        ]
    }
]
```

Figure 1.5: Example request with results filtering

As shown, you can also use the *"false"* string to ommit a sub-result. Please note that the *answers* subtype only accepts *"false"* as a fields selection, since it has to few fields for this feature to make sense.
Finally, please note that the *id* field will always be returned, since the framework needs it.