User Actions

All request URLs in the document are relative to the application root. E.g. if the web application is deployed at http://10.0.0.10:8080/api and this document says to request `/user`, the full URL would be http://10.0.0.10:8080/api/user

Store user actions

User's actions are usually tracked by external publication system. Currently we support one kind of action - which may be considered equivalent to "User has read" or "User is interested in". Depending on specifics of the content and audience, different systems may opt for different schema of registering user's actions - for example, user may be considered interested in an article if they commented on it or scrolled to second screen, or a combination of these.

Relation is established by a call like this one:

```
POST /user?key=<api_key>&userid=<user_id>&contentid=<existing_article_id>
```

where "recommend" is the name of the installed recommendation application.

If relation was established successfully, a similar result would be returned:

```
{
    "version": "Ontotext Recommendations API",
    "type": "SUCCESS",
    "status": "OK",
    "message": "User updated: id = user-1"
}
```

Query parameters:

- **userid** (required) - the user reading an article
- **contentid** (required) - the article a user is reading
- **timestamp** (optional) - the time the action occurred. If not specified, the action occurs "now"

Obtain a list of user's reads

This is done by such call:

```
GET recommend/user/?key=<api_key>&userid=<user_id>
```

Result would be something like this:

```
{
    "version": "Ontotext Recommendations API",
    "type": "USER_INFO",
    "status": "OK",
    "response": {
        "numberOfReads": 4,
        "articles": [
            {
                "id": "doc-en-1",
                "title": "Hello London!",
                "score": 0.8853250741958618,
                "published": "2014-10-14T17:00:37Z",
                "url": "http://example.org/hello-london-article-1",
                "popularity": 3
            },
            {
                "id": "doc-en-2",
                "title": "Hello London Again!",
                "score": 0.8853250741958618,
                "published": "2014-10-15T17:00:37Z",
                "url": "http://example.org/hello-london-article-2",
                "popularity": 1
            }
        ]
    }
}
```


User with ID of "user-1" has read article with ID of "doc-en-1" three times and article of ID of "doc-en-2" once.
As a result, we have **numberOfReads**=4 for the user and also **popularity**=3 and **popularity**=1 for the two articles. Popularity is global, it does not depend on the user we are asking about.

Obtain a count of user reads

Use calls like these:

```
GET /usagestats/userreads/?key=<api_key>&from=aweekago
GET /usagestats/userreads/?key=<api_key>&from=aweekago&to=2
GET /usagestats/userreads/?key=<api_key>&from=10&to=2
GET /usagestats/userreads/?key=<api_key>&from=10
```

The "**to**" parameter is optional and defaults to present time.
Both "**from**" and "**to**" can take following values:

- "amonthago", "aweekago"
- integers representing number of days

Single number is returned as a result. Note that if an user had read the same article twice, it would be considered two reads.

Let's say you want to build a chart for activity at your news site on daily basis for the last 10 days.

You could use this method to get reads for each day and plot them in a chart:

```
GET /usagestats/userreads/?key=<api_key>&from=30&to=29
GET /usagestats/userreads/?key=<api_key>&from=29&to=28
GET /usagestats/userreads/?key=<api_key>&from=28&to=27
...
```

Obtaining statistics on most active users

Another useful method is:

```
GET /usagestats/mostactiveusers/?key=<api_key>&from=aweekago&count=5
```

This method has the same "**from**" and "**to**" parameters as usagestats/userreads and returns top users with most reads. The "**count**" parameter shows size of the list to be returned.
The method has also these optional parameters:

- **minActivity**
- **maxActivity**
- **uniqueArticles** - true/false; Shows if unique articles only are to be counted