

PROJECT DOCUMENTATION

Documentation for URL shortener application

Dolores Škugor

Zagreb, 26.10.2018.

1. Intro

Assignment given by Infobip was to make HTTP service that shorten URLs.

It was divided into smaller tasks:

1. opening of accounts
2. registration of URLs
3. retrieval of statistics
4. statistics page
5. redirecting

Use case

The application is made in Java with Spring Boot framework. This way it was faster to configure Spring project and to maintain focus on developing functionalities required for the assignment.

This type of application is interesting because every letter costs. As Infobip sends a lot of sms messages in its business it is important for them that those messages are shorter and this application will let them do that.

2. Guides for installation and running the application

To checkout project go to following GIT repository:

```
https://github.com/doky91/urlshortener
```

Use the Download button or use the local terminal to checkout the project.

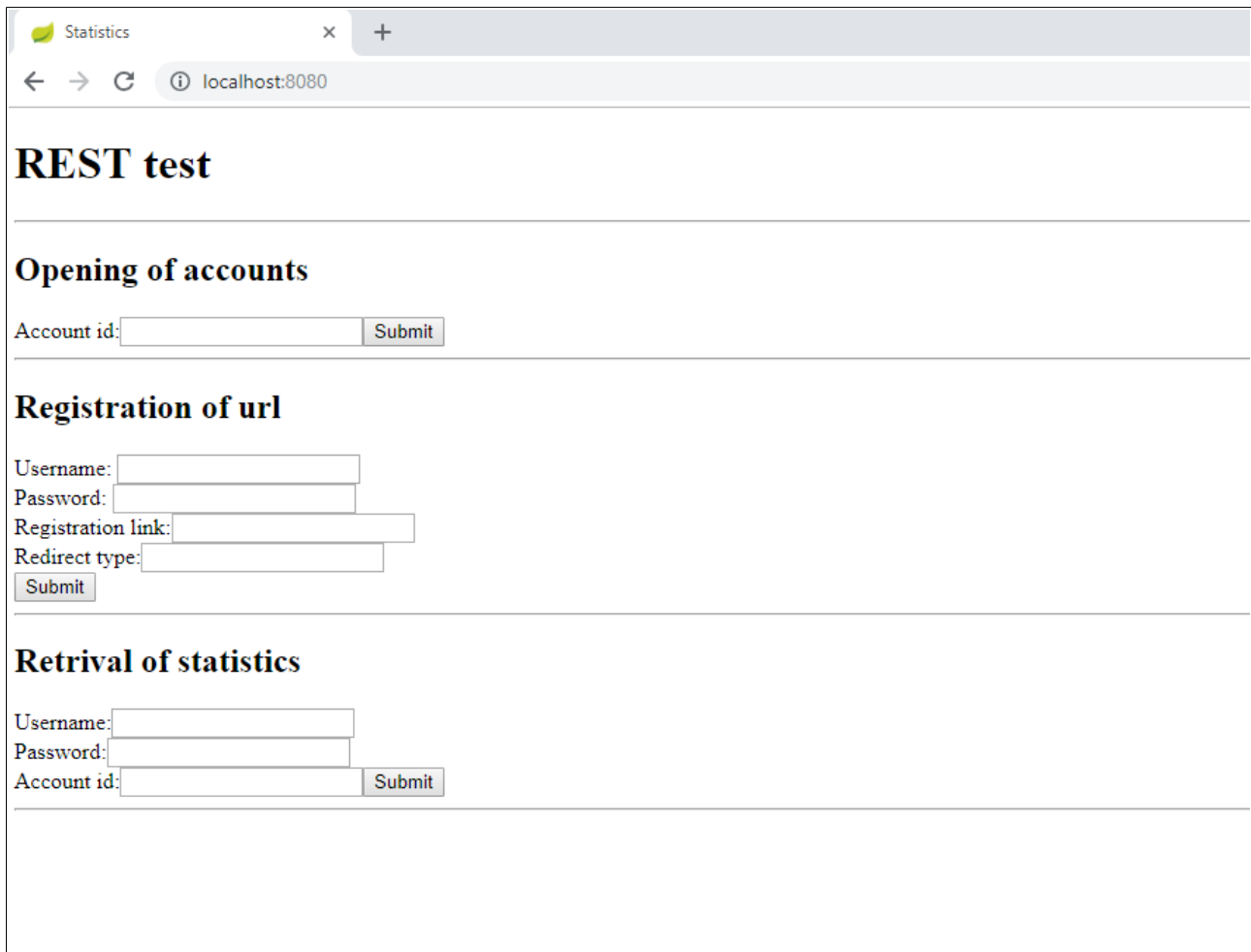
After downloading it is recommended to open some IDE (e.g STS or Eclipse) and import project as *existing maven project* into workplace. After performing import you should run *Update maven project* action. And the final step is to run applicator as *Java application* and select *UrlshortenerApplication* from package `hr.dskugor.urlshortener.urlshortener` as main class.

3. Testing application

After the application is up and running it is possible to test it in two ways:

- by visiting <http://localhost:8080> and using forms
- by using some kind of POST emulator like Postman

Visiting <http://localhost:8080> the following forms are shown:



The screenshot shows a web browser window with the title "Statistics". The address bar displays "localhost:8080". The main content area is titled "REST test" and contains three distinct sections, each with a heading and a form:

- Opening of accounts**: A form with a single input field labeled "Account id:" and a "Submit" button.
- Registration of url**: A form with four input fields labeled "Username:", "Password:", "Registration link:", and "Redirect type:", followed by a "Submit" button.
- Retrival of statistics**: A form with three input fields labeled "Username:", "Password:", and "Account id:", followed by a "Submit" button.

It is important to note that none of the frontend validations were implemented because the focus of the assignment was to backend development.

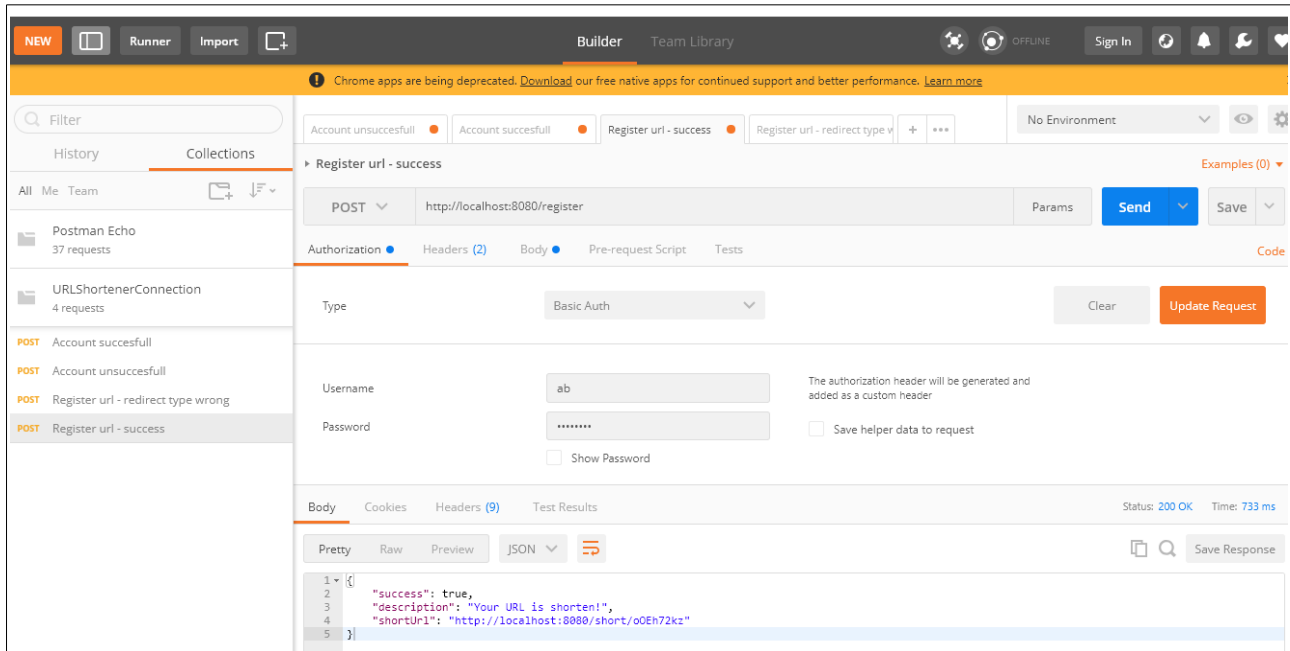
In order for everything to work properly you need to register user to get his password.

After that, it is possible to register link using earlier created user credentials.

For more information about services see chapter about REST documentaion.

Testing using Postman

Using Postman provides you mock service for the requests. You can import file provided or you can manually type the request. In both cases, before you send the request you need to enter credentials. But if you manually type the request you need select basic authentication for routes /register and /statistic and if you import this option will already be selected for you.



4. REST Documentation

Account endpoint

- Allowed methods: POST
- URI: /account
- Request Type: application/json
- Request Headers: none
- Request Body: JSON object
- Response Type: application/json
- Response Body: JSON object

Request body example:

```
{
  accountId: "ab"
}
```

Response success example:

```
{
  success: "true",
  description: "Your account is opened",
  password: "xC345Fc0"
}
```

Response failure example:

```
{
  success: false,
  description: "User with ID : ab already exists!",
  password: null
}
```

Other possible failure is:

- "No valid account id provided!"

Registration endpoint

- Allowed methods: POST
- URI: /register
- Request Type: application/json
- Request Headers: Authorization header with Basic authentication token
- Request Body: JSON object
- Response Type: application/json
- Response Body: JSON object

Request body example:

```
{
  url: "https://www.ebay.com/",
  redirectType: 301
}
```

*redirect type is not mandatory, default is 302

*supported redirect types are 301 and 302

Response success example:

```
{
  success: true,
  description: "Your URL is shorten!",
  shortUrl: "http://localhost:8080/short/TlOI1x2w"
}
```

Response failure example:

```
{
  success: false,
  description: "Redirect type should be 301 or 302!",
  shortUrl: null
}
```

Other possible failure is:

- "No valid url provided!"
- 401 exception in case of unauthorized requests

Statistics endpoint

- Allowed methods: GET
- URI: */statistic/{AccountId}*
- Request Type: GET
- Request Headers: Authorization header with Basic authentication token
- Request Body: none
- Response Type: application/json
- Response Body: JSON object formatted like key:value map

Response success example:

```
{  
  "http://localhost:8080/short/TlOI1X2w": 2  
}
```

Response failure example:

```
{  
  "timestamp": "2018-10-25T20:01:07.248+0000",  
  "status": 401,  
  "error": "Unauthorized",  
  "message": "Unauthorized",  
  "path": "/statistic/abc"  
}
```