# Exercises: HTTP Protocol

Problems for exercises and homework for the [“Java Web Development Basics” course @ SoftUni](https://softuni.bg/trainings/1844/java-web-development-basics-january-2018/internal). Submit your solutions on the **course page** of the **current instance**.

## Parsing HTTP Requests

Implement a simple java application which parses HTTP Requests and returns an appropriate response based on the Request.

You will receive an input of several lines. The **first input line** will contain several **URL paths**, separated by **spaces**.

**Example**: /path /register /login /products/create /admin/users/all

You should store these paths, they are the **valid** **URL**s.

The next **several input lines** will contain **information** about a simple **HTTP Request**.

**Example**:

POST /url HTTP/1.1  
Date: 17/01/2019  
Host: localhost:8000  
Content-Type: application/xml  
Authorization: Basic UGVzaG8=  
  
name=Yum&quantity=50&price=10

You must implement a simple **parser**, which parses **specific** information, from the given **request data**, and returns a well-formatted **HTTP Response** in **text format**.

You must process the **Request Line**.

* Check if the **URL** is present in the **valid** **URLs**

You may be given any header, but you must only process the **Date**, **Host**, **Content-Type** headers.

* **Attach** the **headers** to the **HTTP Response**’s **headers**
* If any of the headers is missing you **don’t need** to do anything.
* Skip the **Authorization** header for this

You must also process the **Request Body**.

* Split the **body parameters**

In the end you should return a Response which contains the processed data from the request in an aggregated format.

HTTP/1.1 200 OK  
Date: 17/01/2019  
Host: localhost:8000  
Content-Type: application/xml  
  
Greetings Pesho! You have successfully created Yum with quantity – 50, price – 10.

As you can see the **Response**’s **Headers** are the same as the **Request**’s **Headers**. The Response Body is the only new thing. It is created in the following format.

Greetings {username}! You have successfully created {firstRequestBodyParameterValue} with {secondRequestBodyParameterName} – {secondRequestBodyParameterValue}, {thirdRequestBodyParameterName} – {thirdRequestBodyParameterValue}.

The username is extracted by **decoding** with **Binary64Decoder** the **Authorization** **Header’s** **value** (after the Basic credential key).

**Example**: Authorization: Basic UGVzaG8= -> Pesho

Then you must format the Request’s body parameters and place them in the Response content.

**NOTE**: If the **URL** is **invalid** (**not present** in **valid URL**s **given** on the **first line**), you should return an HTTP Response in the same format but this time with:

* Status - **404** **Not Found**
* Response Body – "**The requested functionality was not found."**

**NOTE**: If the **Authorization** header is **missing**, you should return an HTTP Response in the same format, but with:

* Status - **401** **Unauthorized**
* Response Body – "**You are not authorized to access the requested functionality."**

**NOTE**: If the Request’s Method is POST and there is NO body parameters, you should return an HTTP Response in the same format but with:

* Status – **400** **Unauthorized**
* Response Body – "**There was an error with the requested functionality due to malformed request."**

If the **Request**’s **Method** is **GET** just print "Greetings {username}!" as Response **body**.

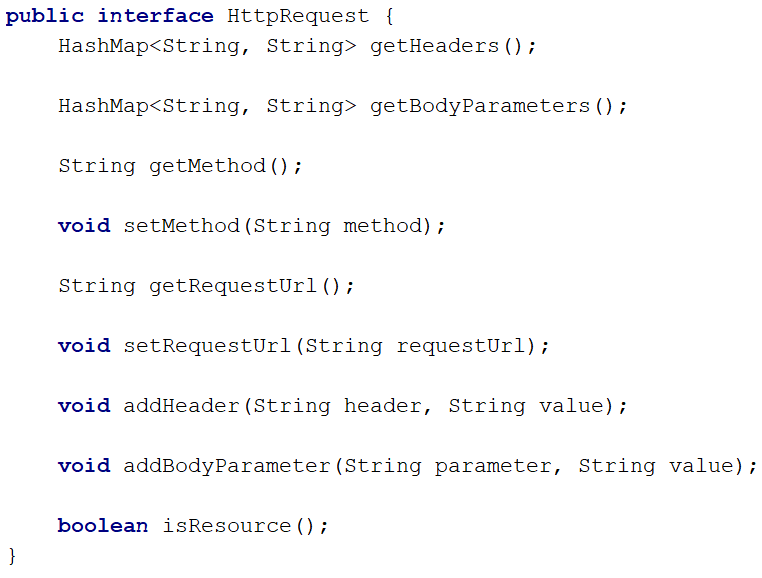
### Example

|  |  |
| --- | --- |
| **Input** | **Output** |
| /url /login /register POST /url HTTP/1.1  Date: 17/01/2019  Host: localhost:8000  Content-Type: application/xml  Authorization: Basic UGVzaG8=  name=Yum&quantity=50&price=10 | HTTP/1.1 200 OK  Date: 17/01/2019  Host: localhost:8000  Content-Type: application/xml  Greetings Pesho! You have successfully created Yum with quantity – 50, price – 10. |
| /url /login /register POST /url HTTP/1.1  Date: 17/01/2019  Host: localhost:8000  name=Yum&quantity=50&price=10 | HTTP/1.1 401 Unauthorized  Date: 17/01/2019  Host: localhost:8000  You are not authorized to access the requested functionality. |
| /create /update  POST /url HTTP/1.1  Host: localhost:8000  Authorization: Basic UGVzaG8=  name=Yum&quantity=50&price=10 | HTTP/1.1 404 Not Found  Date: 17/01/2019  Host: localhost:8000  The requested functionality was not found. |
| /url /update  POST /url HTTP/1.1  Host: localhost:8000  Authorization: Basic UGVzaG8= | HTTP/1.1 400 Bad Request  Date: 17/01/2019  Host: localhost:8000  There was an error with the requested functionality due to malformed request. |

## Create classes

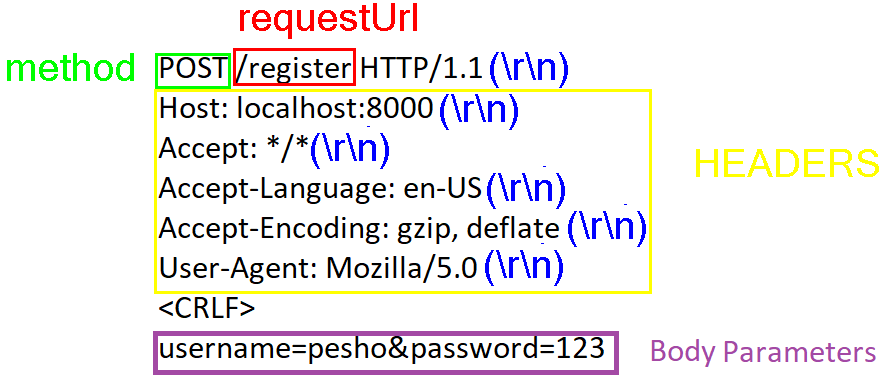
Now, all the parsing logic from the previous task should be aggregated into 2 classes.

The HttpRequest:



Create a class which implements this interface, and write the logic behind the methods. The class should have a method, a requestUrl, a collection of headers and a collection of bodyParameters.

Remember, a request looks like this:

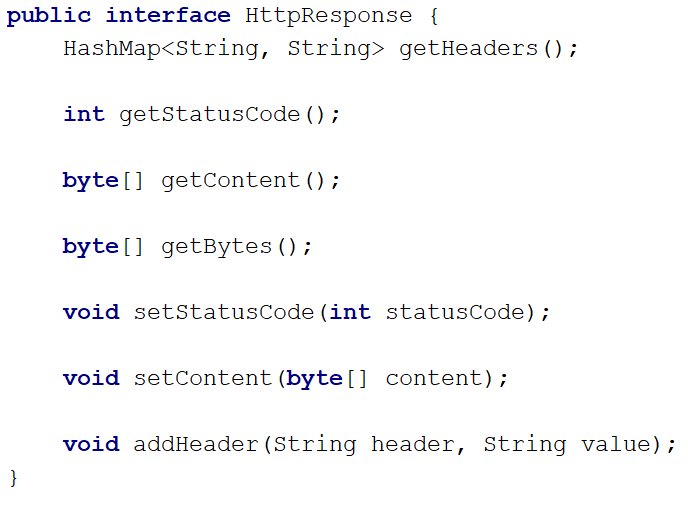


You should receive the **string** in the **constructor** of theRequestclass, exactly as shown above, with **every line**, **delimitered** by “\r\n”. **Everything** should be **parsed** and **formatted** **INSIDE** the class.

The isResource() method should **check** if the requestedUrl is a **resource** and **not an actual route**, and should return a **boolean result**.

### HttpResponse

And the HttpResponse:



Same as the HttpRequest above, you should implement this class, so that it **corresponds** to the **behaviour** defined by the **interface**.

The getBytes() method should **return** the **whole response** (**ResponseLine** + **Headers** + **Content**) as byte array.