



HttpServer

DeepDive Week 1 Day 0



Материалы и Ресурсы

Сегодня мы начнем погружение в тему HTTP. Для того чтобы выполнить сегодняшнее задание необходимо разобраться в следующих темах:

- что такое [сетевой Socket](#)
- что такое HTTP (и уметь находить нужную информацию с [спецификации HTTP/1.1](#))

Все эти знания помогут в реализации нашего собственного HTTP сервиса поверх простого Java I/O Socket (само собой в связке с нашим StdBufferReader ;)).

Для этого мы реализуем два класса:

- HttpRequestHandler - будет отвечать за обработку одного HTTP запроса
- HttpServer - будет отвечать за работу сервера в целом

Важно: можно использовать только импорты которые тут указаны.

HttpRequestHandler

```
package academy.kovalevskiy.javadeepdive.week1.day0;
import academy.kovalevskiy.javadeepdive.week0.day0.StdBufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.net.Socket;

public class HttpRequestHandler {
    // TODO
    public HttpRequestHandler(Socket socket) {
        // TODO
    }
    public void executeRequest() {
        // TODO
    }
}
```

Данный класс должен парсить HTTP запрос и выводить на stdout (в любой форме):

- HTTP метод
- путь
- версию HTTP
- хост

А также сформировать HTTP ответ (как минимум):

- с хедерами:
 - Content-Length
 - Content-Type
- и телом:
 - “It works!”

Как только этот класс сделан можно перейти к HttpServer.

HttpServer

Данный класс имеет следующую сигнатуру:

```
package academy.kovalevskyi.javadeepdive.week1.day0;
import java.io.IOException;
import java.net.ServerSocket;
import java.util.Scanner;

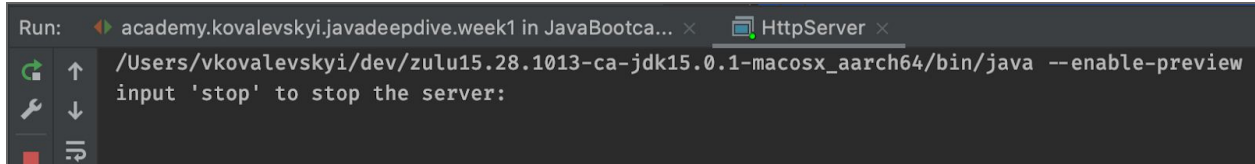
public class HttpServer implements Runnable {
    // TODO
    public static void main(String[] args) {
        // TODO
    }
    @Override
    public void run() {
        // TODO
    }
    public void stop() {
        // TODO
    }
    public boolean isLive() {
        // TODO
    }
}
```

В данном классе нужно реализовать:

- **run** - в вечном цикле ожидает новых HTTP запросов (на localhost:8080)
- **stop** - прерывает цикл ожидания запросов и останавливает сервер
- **isLive** - возвращает статус сервера, запущен он или нет
- **main** - запускает в фоновом потоке сервер и ожидает с stdin слово “stop”, получив которое он останавливает сервер

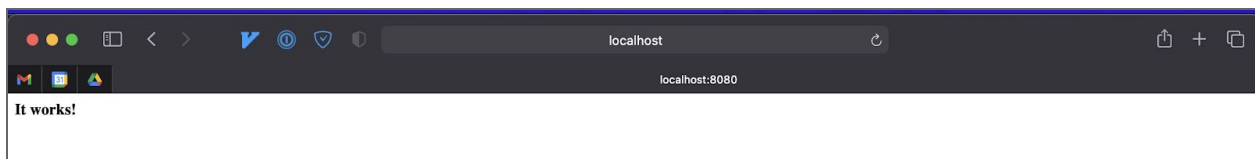
Пример

Пример запуска сервера:

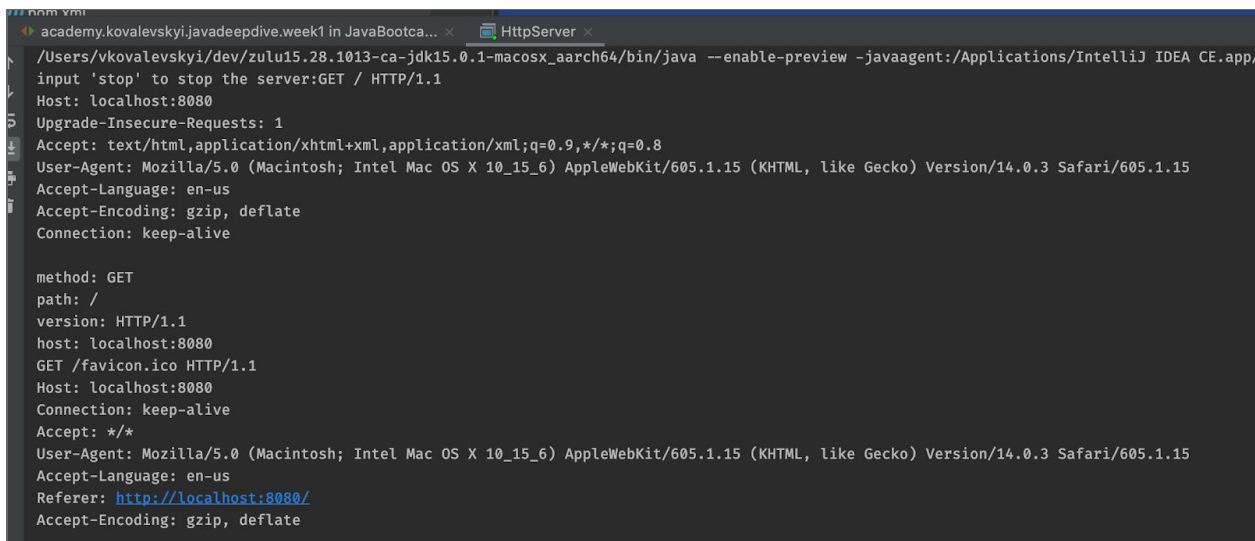


```
Run: academy.kovalevskiy.javadeepdive.week1 in JavaBootca... x HttpServer x
/Users/vkovalevskiy/dev/zulu15.28.1013-ca-jdk15.0.1-macosx_aarch64/bin/java --enable-preview
input 'stop' to stop the server:
```

В этот момент можно открыть браузер и проверить работу сервера:



При вызове сервера из браузера в консоли должно быть видно данные запросов:



```
academy.kovalevskiy.javadeepdive.week1 in JavaBootca... x HttpServer x
/Users/vkovalevskiy/dev/zulu15.28.1013-ca-jdk15.0.1-macosx_aarch64/bin/java --enable-preview -javaagent:/Applications/IntelliJ IDEA CE.app,
input 'stop' to stop the server:GET / HTTP/1.1
Host: localhost:8080
Upgrade-Insecure-Requests: 1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_6) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/14.0.3 Safari/605.1.15
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Connection: keep-alive

method: GET
path: /
version: HTTP/1.1
host: localhost:8080
GET /favicon.ico HTTP/1.1
Host: localhost:8080
Connection: keep-alive
Accept: */*
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_6) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/14.0.3 Safari/605.1.15
Accept-Language: en-us
Referer: http://localhost:8080/
Accept-Encoding: gzip, deflate
```