

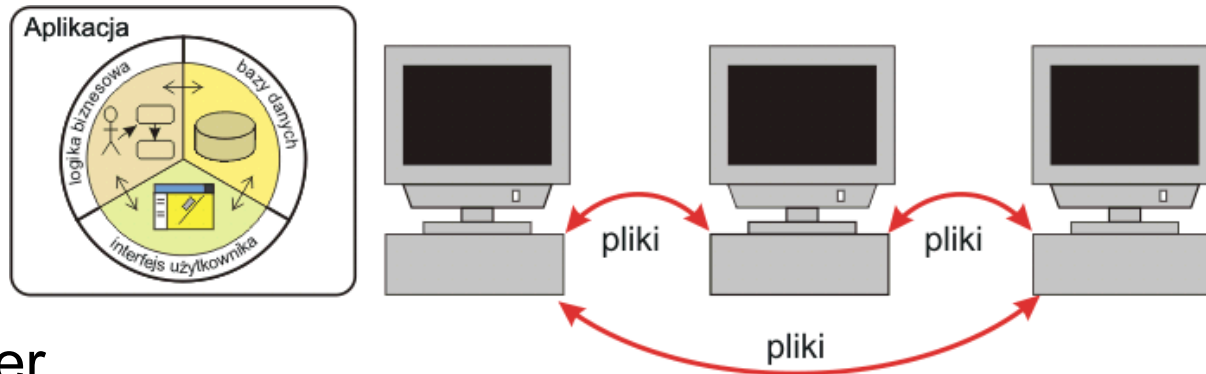
Języki Programowania

dr inż. Tomasz Kubik

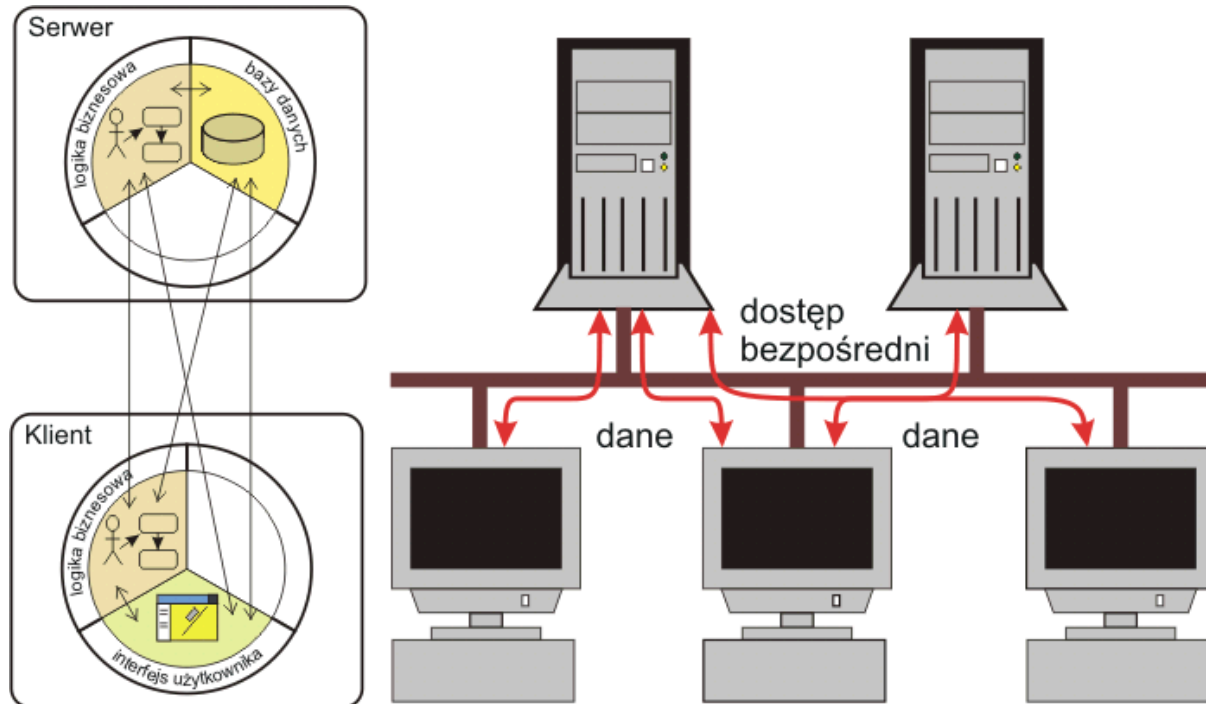
tomasz.kubik.staff.iiar.pwr.edu.pl

Architektura rozwiązań

desktop

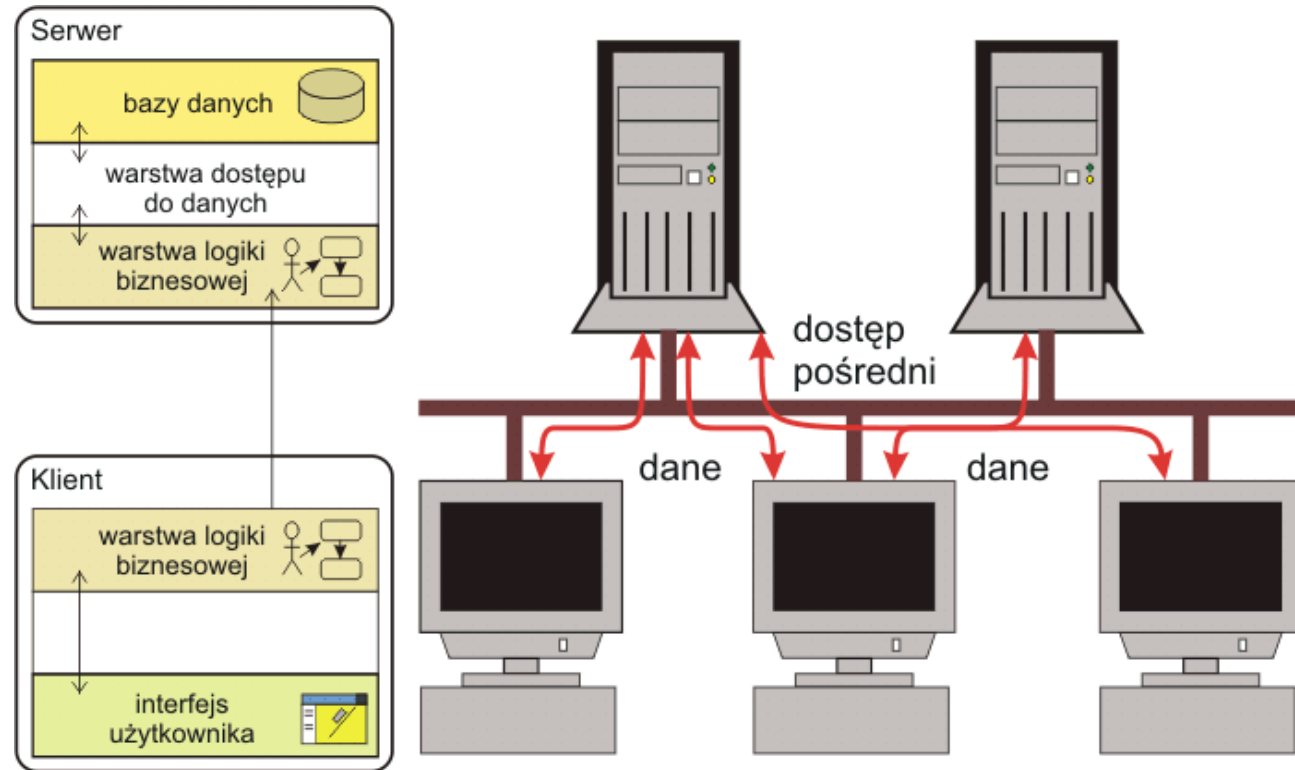


klient-serwer



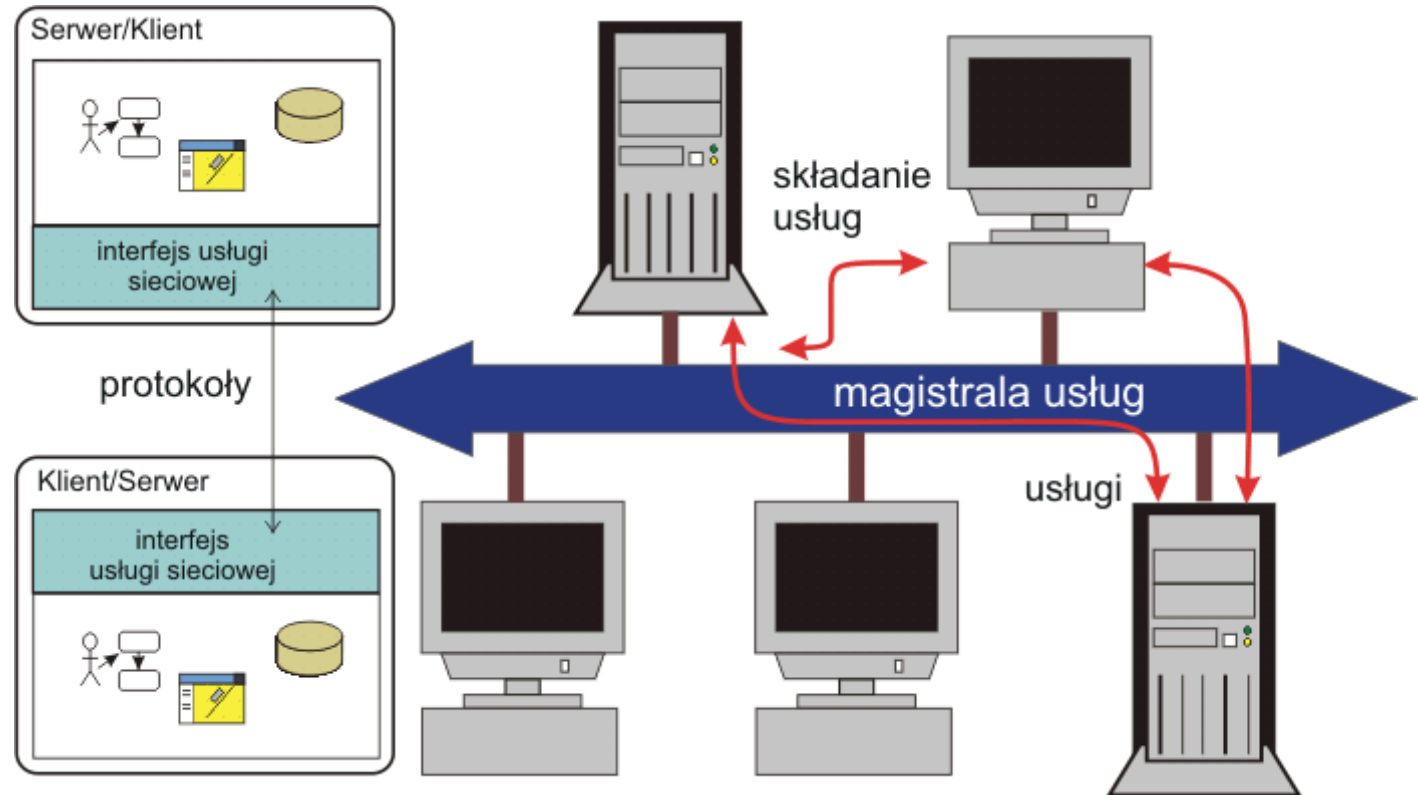
Architektura rozwiązań

n-warstwowa



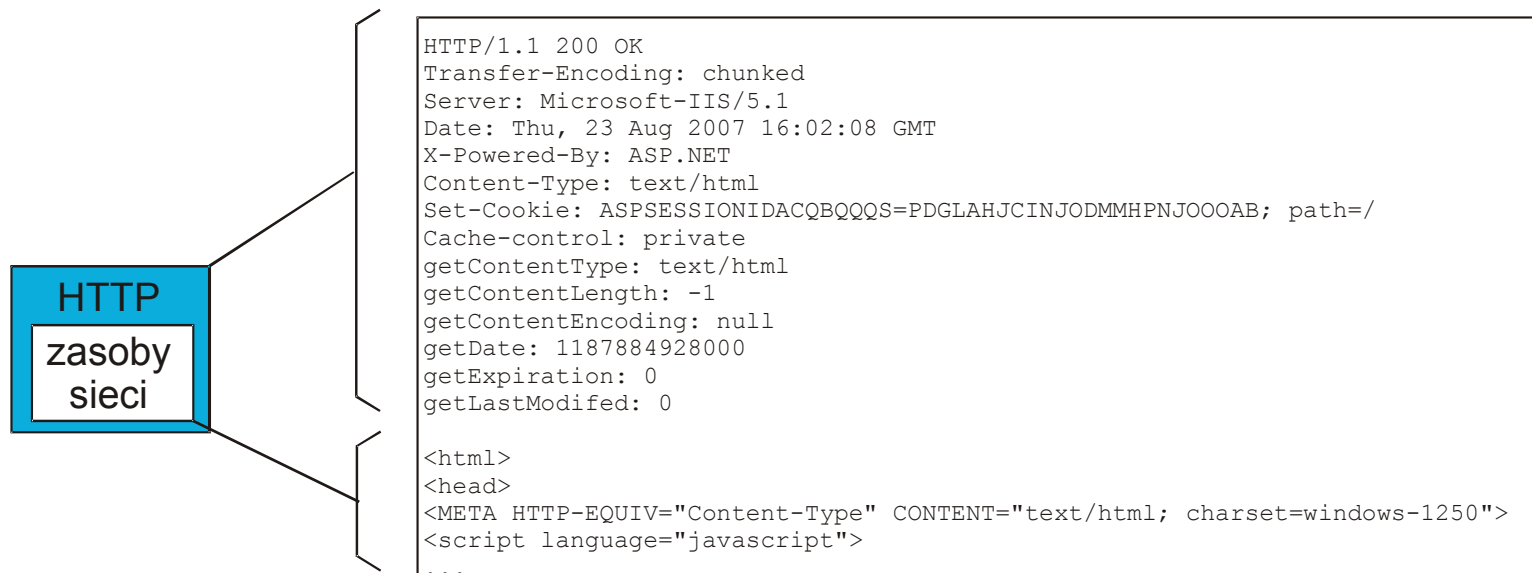
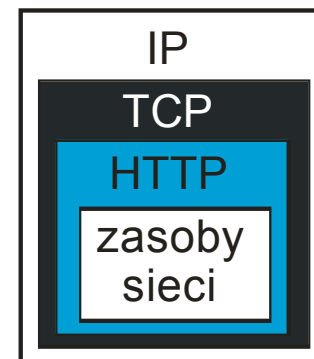
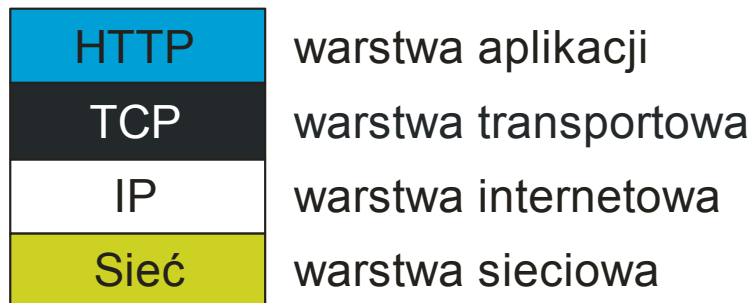
Architektura rozwiązań

zorientowana na usługi



Czterowarstwowy model protokołów sieciowych

- Model „klasyczny”
 - (ang. *TCP/IP Reference Model*)
- Model zagnieżdżony



Do poczytania:

<https://www.studytonight.com/computer-networks/tcp-ip-reference-model>

<https://www.javatpoint.com/computer-network-tcp-ip-model>

<https://www.tutorialspoint.com/The-TCP-IP-Reference-Model>

<https://www.guru99.com/difference-tcp-ip-vs-osi-model.html>

Metody protokołu HTTP

- HTTP/1.1 (RFC 2616)
 - CONNECT, DELETE, **GET**, HEAD, OPTIONS, **POST**, PUT, TRACE, PATCH (RFC 5789)
- HTTP/1.0
 - DELETE, **GET**, HEAD, LINK, **POST**, PUT, UNLINK (zaimplementowane przez niektóre serwery i klientów, ale niespójnie lub w ogóle niezdefiniowane)

[RFC 2616](#)

HTTP/1.1

June 1999

14 Header Field Definitions

This section defines the syntax and semantics of all standard HTTP/1.1 header fields. For entity-header fields, both sender and recipient refer to either the client or the server, depending on who sends and who receives the entity.

14.1 Accept

The Accept request-header field can be used to specify certain media types which are acceptable for the response. Accept headers can be used to indicate that the request is specifically limited to a small set of desired types, as in the case of a request for an in-line image.

```
Accept          = "Accept" ":"
                  #( media-range [ accept-params ] )

media-range     = ( "*"/*
                  | ( type "/" "*" )
                  | ( type "/" subtype )
                  ) * ( ";" parameter )
accept-params   = ";" "q" "=" qvalue *( accept-extension )
accept-extension = ";" token [ "=" ( token | quoted-string ) ]
```

[\[RFC Home\]](#) [\[TEXT\]](#) [\[PDF\]](#) [\[PS\]](#) [\[PDF\]](#) [\[HTML\]](#) [\[Tracker\]](#) [\[IPR\]](#) [\[Errata\]](#) [\[Info page\]](#)

Obsoleted by: [7230](#), [7231](#), [7232](#), [7233](#), [7234](#), [7235](#)
Updated by: [2817](#), [5785](#), [6266](#), [6585](#)
Network Working Group
Request for Comments: 2616
Obsoletes: [2068](#)
Category: Standards Track

DRAFT STANDARD
[Errata Exist](#)

R. Fielding
UC Irvine
J. Gettys
Compaq/W3C
J. Mogul
Compaq
H. Frystyk
W3C/MIT
L. Masinter
Xerox
P. Leach
Microsoft
T. Berners-Lee
W3C/MIT
June 1999

Hypertext Transfer Protocol -- HTTP/1.1

Status of this Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Copyright (C) The Internet Society (1999). All Rights Reserved.

Abstract

The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. It is a generic, stateless, protocol which can be used for many tasks beyond its use for hypertext, such as name servers and distributed object management systems, through extension of its request methods, error codes and headers [\[47\]](#). A feature of HTTP is the typing and negotiation of data representation, allowing systems to be built independently of the data being transferred.

HTTP has been in use by the World-Wide Web global information initiative since 1990. This specification defines the protocol referred to as "HTTP/1.1", and is an update to [RFC 2068](#) [\[33\]](#).

Fielding, et al.

Standards Track

[Page 1]

Do poczytania:

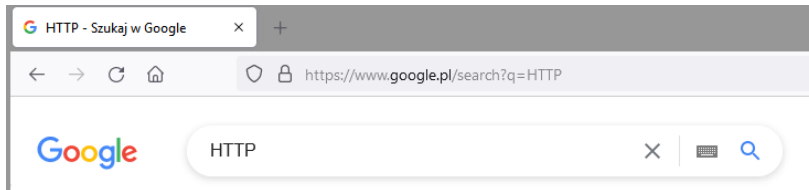
<https://datatracker.ietf.org/doc/html/rfc2616>

<https://www.samouczekprogramisty.pl/protokol-http/>

Najczęściej wykorzystywane metody protokołu HTTP

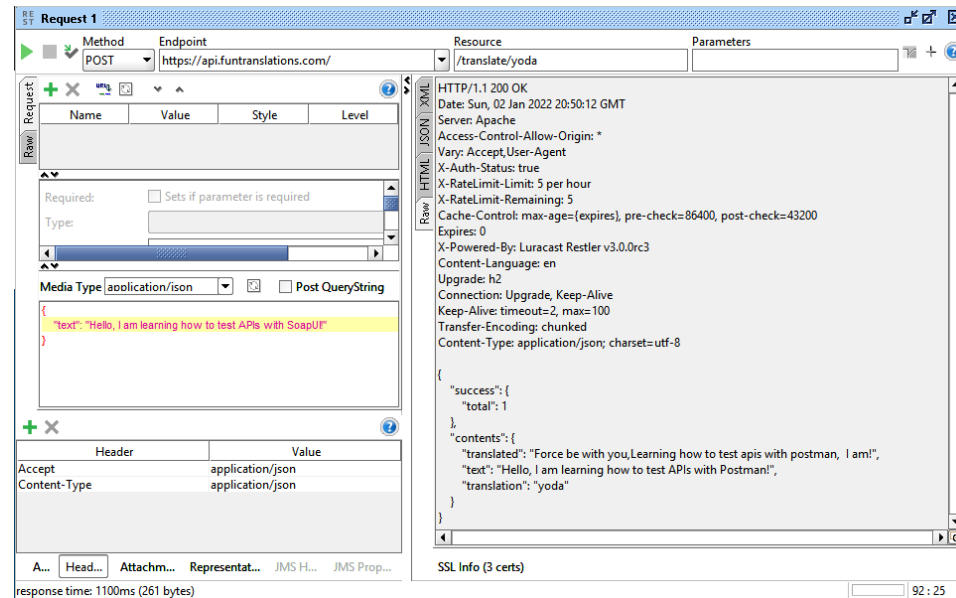
GET

- bywa utożsamiana z ciągiem URL wpisywanym w pasek adresowy przeglądarki lub linkiem do zasobu w dokumencie HTML, jednak **metoda GET nie jest ciągiem URL !!!**
- parametry zapytania są przekazywane w ciągu URL za pomocą par klucz-wartość, jednak nagłówki protokołu HTTP muszą być przekazane w inny sposób,
- ciągi URL mogą podlegać ograniczeniom na długość linii,
- ciąg URL może posłużyć jako zakładka.



POST

- wywoływana z wnętrza aplikacji,
- stosowana, gdy do przekazywania w żądaniach obszernych danych (jako dołączanej zawartości),
- całego zapytania nie da się zapisać w postaci ciągu URL (nie da się zapisać jako zakładkę).



Narzędzia do testowania:

- SoapUI (<https://www.soapui.org/>),
- PostMan (<https://www.postman.com/>),
- ReqBin (<https://reqbin.com/>).



Jeśli nie oznaczono inaczej, ta strona internetowa wraz z zawartością podlegają licencji [Uznanie autorstwa-Użycie niekomercyjne-Na tych samych warunkach 3.0 Polska \(CC BY-NC-SA 3.0 PL\)](https://creativecommons.org/licenses/by-nc-sa/3.0/pl/).

```
<div id="akn" style="padding-right:220px">
  <a class="cc-logo" href="https://creativecommons.org/licenses/by-nc-sa/3.0/pl/" target="_blank" style="position: absolute;
    left: 20px; top: 8px; width: 140px; height: 180px; background: url(/images/cc-by-nc-sa.png) no-repeat left top;"></a>
  <p> Jeśli nie oznaczono inaczej, ta strona internetowa wraz z zawartością podlegają licencji <a
href="https://creativecommons.org/licenses/by-nc-sa/3.0/pl/" rel="license" target="_blank"> Uznanie autorstwa-Użycie niekomercyjne-Na
tych samych warunkach 3.0 Polska (CC BY-NC-SA 3.0 PL)</a>.
</p></div>
```

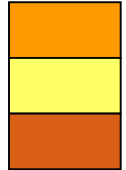
Wywołanie metody GET przez przeglądarkę

- Po wpisaniu URLa w pasek adresowy i jego zatwierdzeniu przeglądarka internetowa wysyła odpowiednio sparametryzowane żądanie GET (patrz nagłówki), a następnie wyświetla odpowiednio sformatowaną zawartość odpowiedzi (patrz content)
 - jeśli strona internetowa jest statyczna, bez żadnych dodatkowych zależności, do pobrania całej jej zawartości wystarcza jedno zapytanie,
 - obecnie większość stron internetowych jest mocno rozbudowana (jednodokumentowe strony należą do rzadkości), zwykle więc przeglądarka musi uruchomić „dodatkowe, uzupełniające zapytania” (celem pobrania zależności, obsłużenia przekierowań itp.)



Elementy zapytania HTTP GET

- linia żądania
- nagłówki HTTP
 - nagłówki ogólne (pojawiają się w zapytaniu i odpowiedzi HTTP)
 - nagłówki żądania (są specyficzne dla konkretnego zapytania HTTP)
 - nagłówki encji (pojawiają się, jeśli pojawia się zawartość)



```
GET /request=WMS HTTP/1.1
```

```
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shock  
wave-flash, application/vnd.ms-excel, application/vnd.ms-powerpoint, application  
/mword, */*
```

```
Accept-Language: pl
```

```
Accept-Encoding: gzip, deflate
```

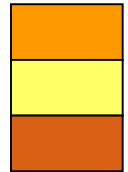
```
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1  
.4322)
```

```
Host: localhost:3541
```

```
Connection: Keep-Alive
```

Elementy zapytania HTTP POST

- linia żądania
- nagłówki HTTP
 - nagłówki ogólne (pojawiają się w zapytaniu i odpowiedzi HTTP)
 - nagłówki żądania (są specyficzne dla konkretnego zapytania HTTP)
 - nagłówki encji (pojawiają się, jeśli pojawia się zawartość)
- pusta linia (kończąca nagłówki)
- ciało wiadomości



```
POST /search HTTP/1.1
```

```
Host: www.google.com
```

```
User-Agent: Mozilla/5.0 Galeon/1.2.5 (X11; Linux i686; U;) Gecko/20020606
```

```
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,  
text/plain;q=0.8,video/x-mng,image/png,image/jpeg,image/gif;q=0.2,  
text/css,*/*;q=0.1
```

```
Accept-Language: en
```

```
Accept-Encoding: gzip, deflate, compress;q=0.9
```

```
Accept-Charset: ISO-8859-1, utf-8;q=0.66, */*;q=0.66
```

```
Keep-Alive: 300
```






```
Connection: keep-alive
```

```
Content-Type: application/x-www-form-urlencoded
```

```
Content-Length: 31
```

```
hl=en&q=HTTP&btnG=Google+Search
```

Elementy odpowiedzi HTTP

- linia statusu (wersja HTTP, kod statusu, krótki opis statusu) 
- nagłówki HTTP
 - nagłówki ogólne (pojawiają się w zapytaniu i odpowiedzi HTTP) 
 - nagłówki odpowiedzi (są specyficzne dla konkretnej odpowiedzi HTTP) 
 - nagłówki encji (pojawiają się, jeśli pojawia się zawartość) 
- pusta linia (kończąca nagłówki)
- ciało wiadomości 

```
HTTP/1.1 302 Object moved
```

```
Server: Microsoft-IIS/5.1
```

```
Date: Thu, 23 Aug 2007 17:48:48 GMT
```

```
X-Powered-By: ASP.NET
```

```
Location: localstart.asp
```

```
Content-Length: 121
```

```
Content-Type: text/html
```

```
Set-Cookie: ASPSESSIONIDACQBQQQS=CEGLAHJCPAEBAIINILNPHKAF; path=/  
Cache-control: private
```

```
<head><title>Object moved</title></head>
```

```
<body><h1>Object Moved</h1>This object may be found <a  
HREF="">here</a>.</body>
```

Podgląd komunikacji HTTP/HTTPS w przeglądarce

- Przeglądarki udostępniają narzędzia dla deweloperów (naciśnij F12 w Mozilla FireFox lub Chrome) pozwalające obserwować przebieg komunikacji.
- Na podglądzie widać, że aby wyświetlić stronę internetową upublicznią pod adresem <https://www.google.com> przeglądarka uruchamia całą sekwencję żądań GET i POST (wynikających z konieczności pobrania zależności, obsługi sesji, obsługi ciasteczek itd.).

The screenshot shows the Google.pl homepage. The developer tools are open, displaying the 'Network' tab. The table below lists the network requests:

Stan	Metoda	Domena	Plik	Typ	Przesłano	Rozmiar	0 ms	2.56 s
200	GET	www.google.pl	/	document	35.50 KB	109.63 KB	142 ms	
200	POST	www.google.pl	gen_204?atyp=i&ei=zVHRYT2MuyWxc8P5sWbqAQ&ct=slh&v=t m=cdos.crdpfsmjs...	html	377 B	0 B	94 ms	
200	POST	www.google.pl	gen_204?i=webhp&t=af&atyp=csi&ei=BvLRYe2JlamUxc8P8l20-/ /13 (beacon)	html	377 B	0 B	59 ms	
200	GET	www.google.pl	client_204?atyp=i&biw=888&bih=466&dpr=1&ei=BvLRYe2JlamUxc8P8l20-/ /3 (img)	html	545 B	0 B	39 ms	
200	POST	www.google.pl	gen_204?atyp=i&r=1&ei=BvLRYe2JlamUxc8P8l20-A0&dt19=2&z m=cdos.crdpfsmjs...	html	377 B	0 B	61 ms	

Summary: 12 żądań, Przesłano: 109.63 KB / 42.62 KB, 1.46 s, DOMContentLoaded: 272 ms, load: 919 ms.

The screenshot shows the Google.com homepage. The developer tools are open, displaying the 'Network' tab. The table below lists the network requests:

Name	Headers	Preview	Response	Initiator	Timing
new-tab-page					
new_tab_page.js					
text_defaults_md.css					
shared_vars.css					
roboto.css					
shared.rolloup.js					
polymer_bundled.min.js					
strings.mjs					
load_time_data.mjs					
main.hindins_lite.js					

Summary: 95 requests, 1.7 MB transferred.

Headers for 'new-tab-page':

- Request URL: chrome://new-tab-page/
- Request Method: GET
- Status Code: 200 OK
- Referrer Policy: strict-origin-when-cross-origin

Response Headers:

- Cache-Control: no-cache
- Content-Security-Policy: child-src https: chrome-untrusted://new-tab-page/object-src 'none'; script-src chrome://resources chrome://test 'self' 'unsafe-inline' https://frame-ancestors 'none';

Gniazda TCP/IP w Java

Dwie główne klasy z pakietu `java.net`:

- `Socket` (gniazdo klienckie) oraz
- `SocketServer` (gniazdo serwerowe)

pozwalają programiście implementującemu warstwę aplikacji w łatwy sposób obsłużyć wymianę danych pomiędzy stronami połączenia, bez wnikania w detale implementacji niższych warstw.

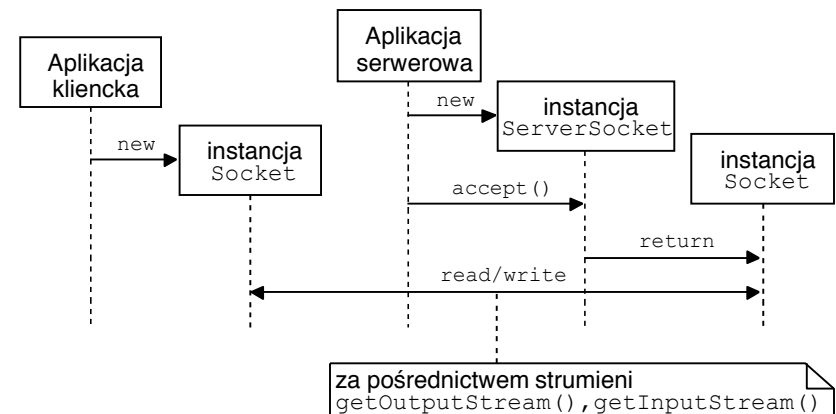
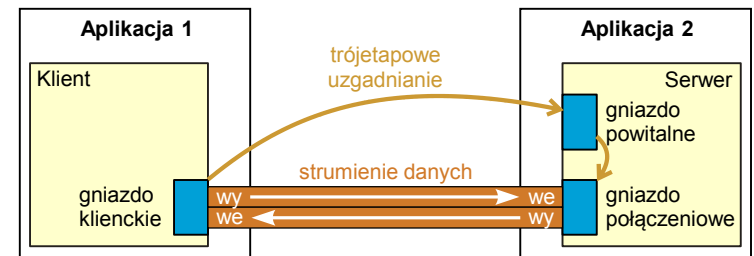
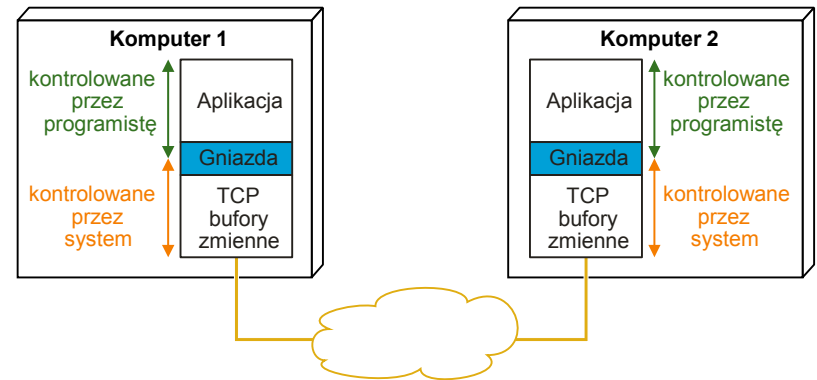
Stronami połączenia są:

- **klient** (wysyłający żądanie) oraz
- **serwer** (przyjmujący żądanie),

przy czym jednak aplikacja może pełnić równocześnie rolę klienta jak i serwera.

Strumień pracy:

- Serwer tworzy obiekt typu `SocketServer`, parametryzując go numerem portu nasłuchowego (gniazdo powitalne) i oczekuje na przychodzące połączenia.
- Klient tworzy obiekt typu `Socket`, parametryzując go adresem IP serwera oraz numerem jego portu nasłuchowego.
- Następnie dochodzi do trójetapowego uzgadniania i nawiązywania połączenia
 - na początek klient zgłasza się pod numer portu nasłuchowego serwera (do gniazda powitalnego),
 - kiedy serwer zauważy tę aktywność, tworzy nowe gniazdo na nowym porcie do komunikacji z tym konkretnym klientem (gniazdo połączeniowe),
 - w końcu dochodzi do nawiązania połączenia TCP między gniazdem klienta a tym nowym gniazdem serwera.
- Po nawiązaniu połączenia wymiana danych odbywa się poprzez pisanie/czytanie do/ze strumieni wyjściowych/wejściowych.



Do poczytania:

http://www2.ic.uff.br/~michael/kr1999/2-application/2_06-sockettcp.htm

<https://www.samouczekprogramisty.pl/protokol-http/>

<https://docs.oracle.com/javase/8/docs/technotes/guides/net/overview/overview.html>

<https://edux.pjwstk.edu.pl/mat/249/lec/KlientSerwer/KlientSerwer.html>

Przykład implementacji (część aplikacji klienckiej)

```
class Sender{
    public void send(String message, String host, int port){
        Socket s;
        try {
            s = new Socket(host,port);
            OutputStream out = s.getOutputStream();
            PrintWriter pw = new PrintWriter(out, false);
            pw.println(message);
            pw.flush();
            pw.close();
            s.close();
        } catch (UnknownHostException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```

Przykład implementacji (część aplikacji serwerowej)

```
interface MyListener {
    void messageReceived(String theLine);
}

class Receiver {
    private List<MyListener> ml = new ArrayList<MyListener>();
    private Thread t = null;
    private int port = 0;
    private ServerSocket s = null;

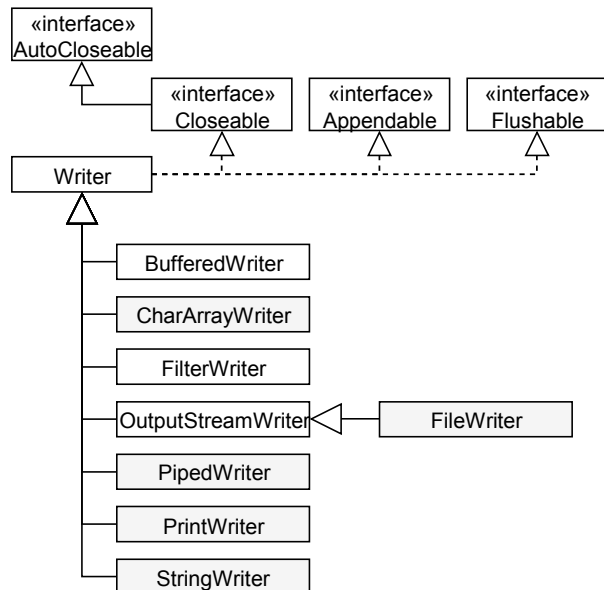
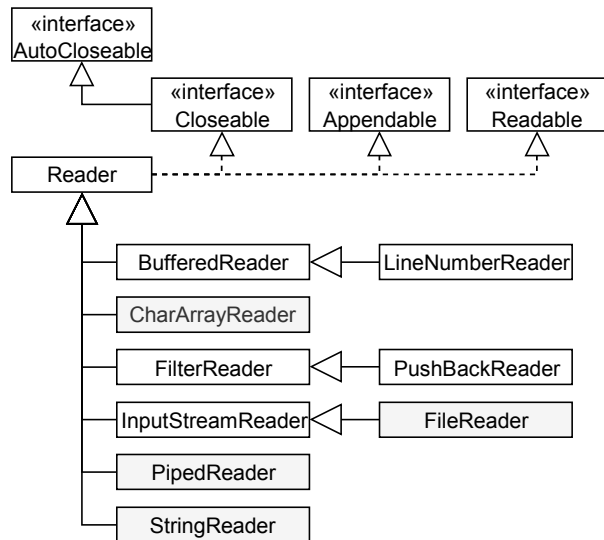
    public void stop() {
        t.interrupt();
        try {
            s.close();
        } catch (IOException e) {}
    }

    public void start() {
        t = new Thread(()->{
            try {
                s = new ServerSocket(port);
                System.out.println("Socket server local port: " +
s.getLocalPort());
                while (true) {
                    Socket sc = s.accept();
                    System.out.println("Socket local port: " +
sc.getPort());
                    InputStream is = sc.getInputStream();
                    InputStreamReader isr = new InputStreamReader(is);
                    BufferedReader br = new BufferedReader(isr);
                    String theLine = br.readLine();
                    ml.forEach((item) ->
item.messageReceived(theLine));
                    sc.close();
                }
            } catch (SocketException e){
                // podczas przerywania wątku metoda accept zgłosi wyjątek
                // z wiadomością: socket closed
            }
            catch (IOException e) {}
        });
        t.start();
    }

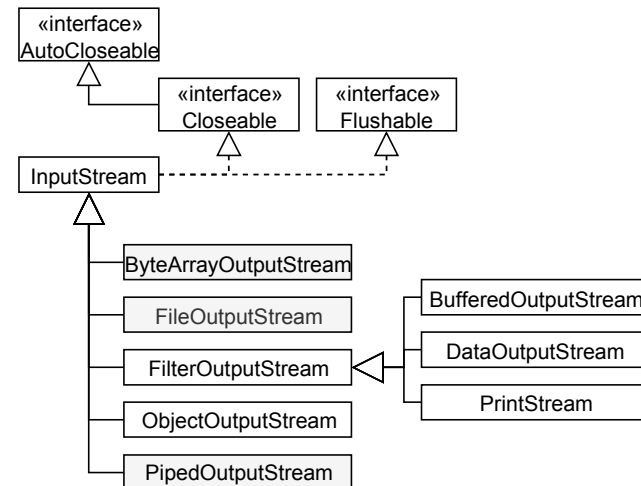
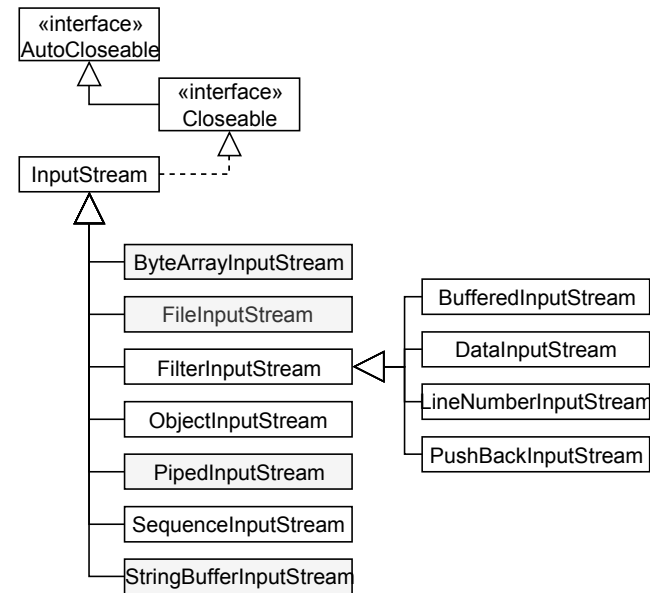
    public void addMyListener(MyListener m) { ml.add(m);
    }
    public void removeMyListener(MyListener m) { ml.remove(m);
    }
    Receiver(int port) { this.port = port;
    }
}
```

Java IO (pakiet java.io)

- Strumienie znaków



- Strumienie bajtów



Do poczytania:

<https://www.javatpoint.com/java-io>

Java IO (przykład)

```
import java.io.*;

public class CopyBytes {
    public static void main(String[] args) throws IOException {
        File inputFile = new File("inFile.txt");
        File outputFile = new File("outFile.txt");

        FileInputStream in = new FileInputStream(inputFile);
        FileOutputStream out = new FileOutputStream(outputFile);
        int c;

        while ((c = in.read()) != -1)
            out.write(c);

        in.close();
        out.close();
    }
}
```

Java NIO (pakiet `java.nio`)

- NIO wprowadzono w jdk 1.4 (pakiet `java.nio`), a później zaktualizowano do NIO.2 w jdk 1.7, wprowadzając m.in. asynchroniczne we/wy (pakiet `java.nio.file` oraz kilka dodatków w `java.nio`)
- NIO oraz NIO.2 to na nowo napisane API (klasy i interfejsy) służące do wykonywania operacji we/wy
 - NIO: selektory (ang. *selectors*) / **wzorzec reaktora**
(<https://github.com/kasun04/nio-reactor>,
<https://programmer.ink/think/three-reactor-models-based-on-nio.html>)
 - NIO.2: obsługa zakończenia (ang. *completion handlers*) / **wzorzec proaktora**
(<https://www.dre.vanderbilt.edu/~schmidt/PDF/proactor.pdf>,
http://didawiki.cli.di.unipi.it/lib/exe/fetch.php/magistraleinformatica/tdp/tpd_reactor_proactor.pdf)

```
import java.io.File;
public class File
extends Object
implements Serializable, Comparable<File>
```

```
file = new File("path/to/file.txt")
file = new File(parentFile, "file.txt")
file.getFileName()
file.getParentFile()
file.mkdirs()
file.length()
file.exists()
file.delete()
new FileOutputStream(file)
new FileInputStream(file)
file.listFiles(filter)
```

```
import java.nio.file.Path;
public interface Path
extends Comparable<Path>, Iterable<Path>, Watchable
```

```
path = Paths.get("path/to/file.txt")
path = parentPath.resolve("file.txt")
path.getFileName().toString()
path.getParent()
Files.createDirectories(path)
Files.size(path)
Files.exists(path)
Files.delete(path)
Files.newOutputStream(path)
Files.newInputStream(path)
Files.list(path).filter(filter).collect(toList())
```

Przykład użycia klas IO oraz NIO

```
void sample() throws IOException {
    // -> IO
    File file = new File("io.txt");

    // -> NIO
    Path path = Paths.get("nio.txt");
    List<String> lines = Arrays.asList(String.valueOf(Calendar.getInstance().getTimeInMillis()),
        "line one",
        "line two");

    // -> IO
    if (file.exists()) {
        // Note: Not atomic
        throw new IOException("File already exists");
    }
    try (FileOutputStream outputStream = new FileOutputStream(file)) {
        for (String line : lines) {
            outputStream.write((line + System.lineSeparator()).
                getBytes(StandardCharsets.UTF_8));
        }
    }

    // -> NIO
    try (OutputStream outputStream = Files.newOutputStream(path,
        StandardOpenOption.CREATE_NEW)) {
        for (String line : lines) {
            outputStream.write((line + System.lineSeparator()).
                getBytes(StandardCharsets.UTF_8));
        }
    }
}
```

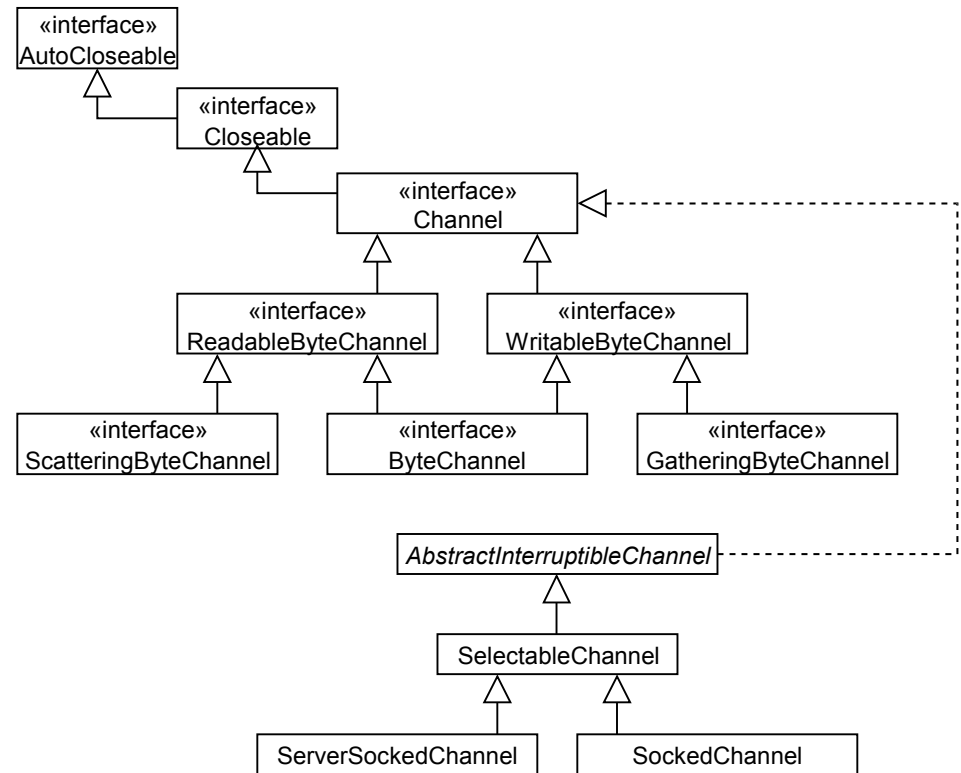
Do poczytania:

<https://riptutorial.com/java/example/7225/migrating-from-java-io-file-to-java-7-nio--java-nio-file-path->

<https://www.baeldung.com/java-nio-selector>

Przykład użycia (java.nio.channels)

```
void sample2() throws IOException {  
    FileInputStream input = new FileInputStream ("inFile.txt");  
    ReadableByteChannel source = input.getChannel();  
    FileOutputStream output = new FileOutputStream ("outFile.txt");  
    WritableByteChannel destination = output.getChannel();  
    ByteBuffer buffer = ByteBuffer.allocateDirect(20 * 1024);  
  
    while (source.read(buffer) != -1) {  
        buffer.flip();  
        while (buffer.hasRemaining()) {  
            destination.write(buffer);  
        }  
        buffer.clear();  
    }  
    source.close();  
    destination.close();  
}
```



Do poczytania:

<https://ducmanhphan.github.io/2020-04-06-Understanding-about-Java-NIO-API-how-to-use-channel-buffer/>

<https://www.javatpoint.com/java-nio-channels>