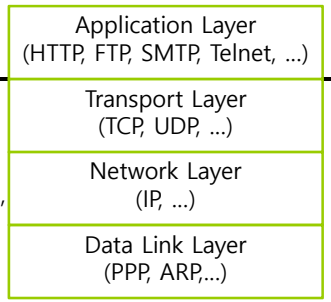
Networking

514770
2018년 가을학기
11/12/2018
박경신

TCP/IP 소개



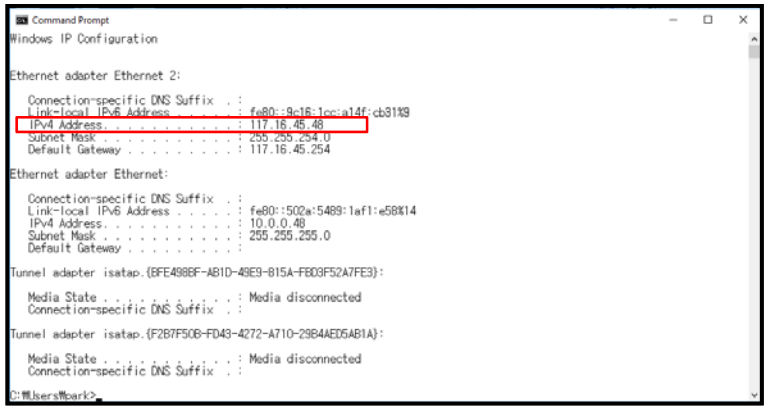
- TCP/IP 프로토콜
  - Application Layer
    - SMTP(Simple Mail Transfer Protocol), Telnet, FTP(File Transfer Protocol), HTTP(HyperText Transfer Protocol) 등
  - Transport Layer
    - TCP(Transmission Control Protocol) 신뢰성 있는 데이터 전송 프로토콜
    - UDP(User Datagram Protocol) 비연결형 서비스 (신뢰성 없음) 데이터 전송 프로토콜
  - Network Layer
    - IP(Internet Protocol) 패킷 교환 네트워크에서 송신 호스트와 수신 호스트가 데이터를 주고 받는 것을 관장하는 프로토콜
  - Data Link Layer
    - OS, 디바이스 드라이버, 네트워크 인터페이스를 제어하는 계층
    - PPP(Point to Point Protocol), ARP(Address Resolution Protocol)

IP 주소

- IP 주소
  - 네트워크 상에서 유일하게 식별될 수 있는 컴퓨터 주소
    - 숫자로 구성된 주소
    - 4개의 숫자가 '.'으로 연결
      - 예) 192.156.11.15
  - 숫자로 된 주소는 기억하기 어려우므로 www.naver.com과 같은 문자열로 구성된 도메인 이름으로 바꿔 사용
    - DNS(Domain Name Server)
      - 문자열로 구성된 도메인 이름을 숫자로 구성된 IP 주소로 자동 변환
  - 현재는 32비트의 IP 버전 4(IPv4)가 사용되고 있음
    - IP 주소 고갈로 인해 128비트의 IP 버전 6(IPv6)이 점점 사용되는 추세

내 컴퓨터의 IP 주소 확인하기

- 내 컴퓨터의 윈도우에서 명령창을 열어 **ipconfig** 명령 수행

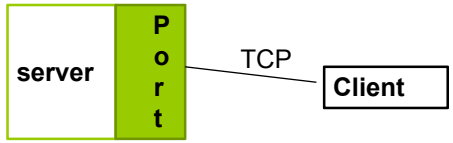


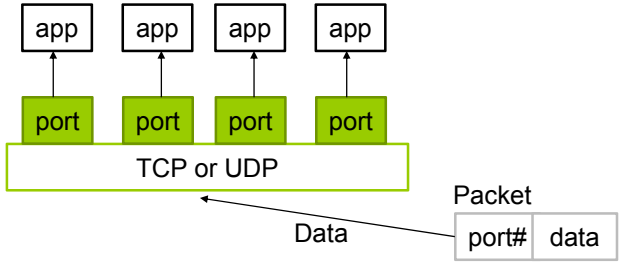
포트 (Port #)

- 포트 (Port)
  - 통신하는 프로그램 간에 가상의 연결단 포트 생성
    - IP 주소는 네트워크 상의 컴퓨터 또는 시스템을 식별하는 주소
    - 포트 번호를 이용하여 통신할 응용프로그램 식별
  - 모든 응용프로그램은 하나 이상의 포트 생성 가능
    - 포트를 이용하여 상대방 응용프로그램과 데이터 교환
  - 잘 알려진 포트(well-know ports)
    - 시스템이 사용하는 포트 번호
    - 잘 알려진 응용프로그램에서 사용하는 포트 번호
      - 0부터 1023 사이의 포트 번호
      - ex) 텔넷 23, HTTP 80, FTP 21
    - 잘 알려진 포트 번호는 개발자가 사용하지 않는 것이 좋음
      - 충돌 가능성 있음

포트를 이용한 통신

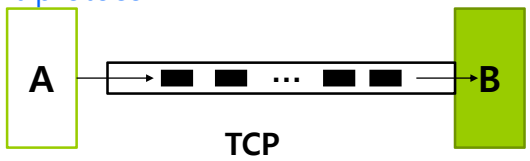
- TCP/UDP 프로토콜은 포트(port)를 이용해서 통신할 응용프로그램을 연결함



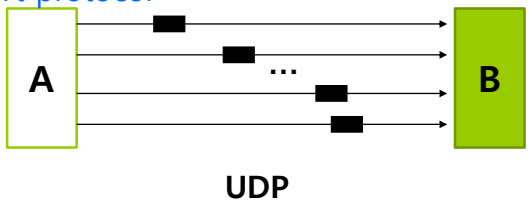


TCP vs UDP

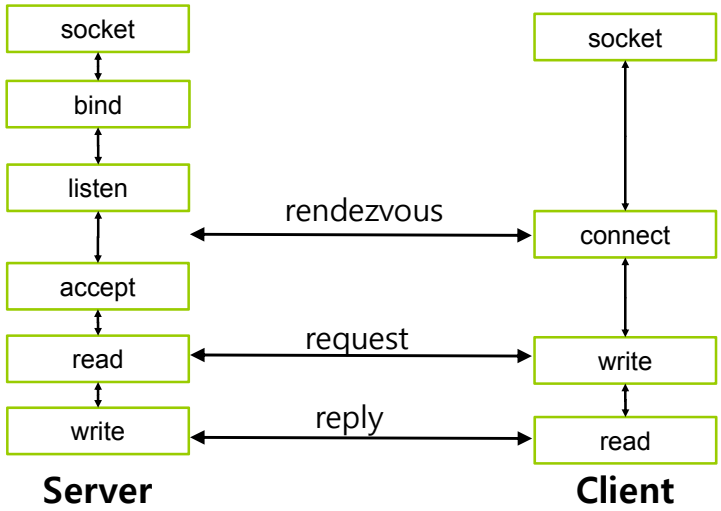
- **TCP** is a **connection-oriented reliable stream** transport protocol



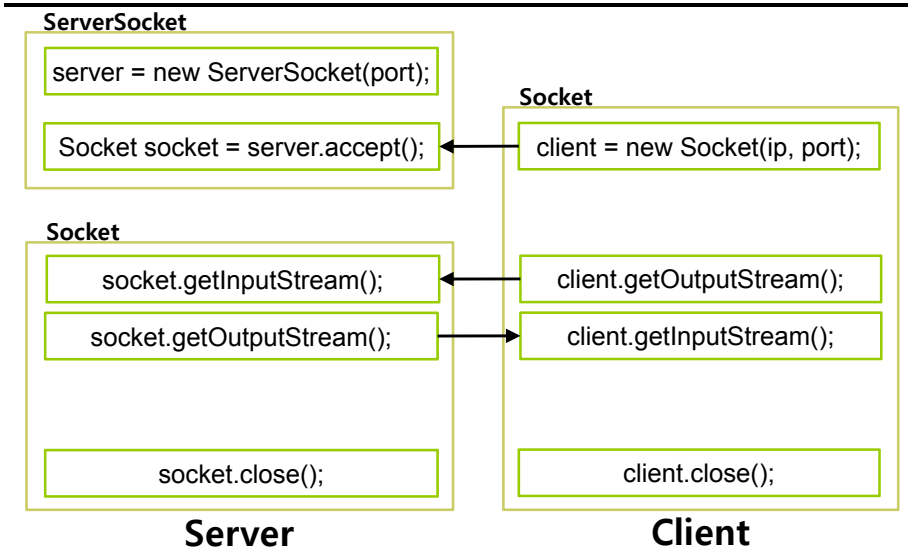
- **UDP** is a **connectionless unreliable datagram** transport protocol



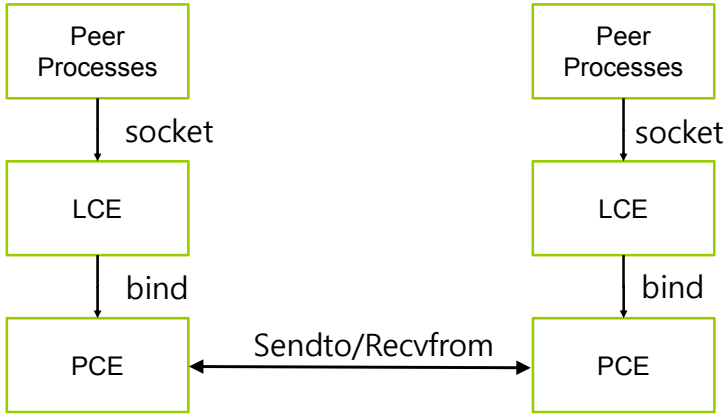
TCP Socket



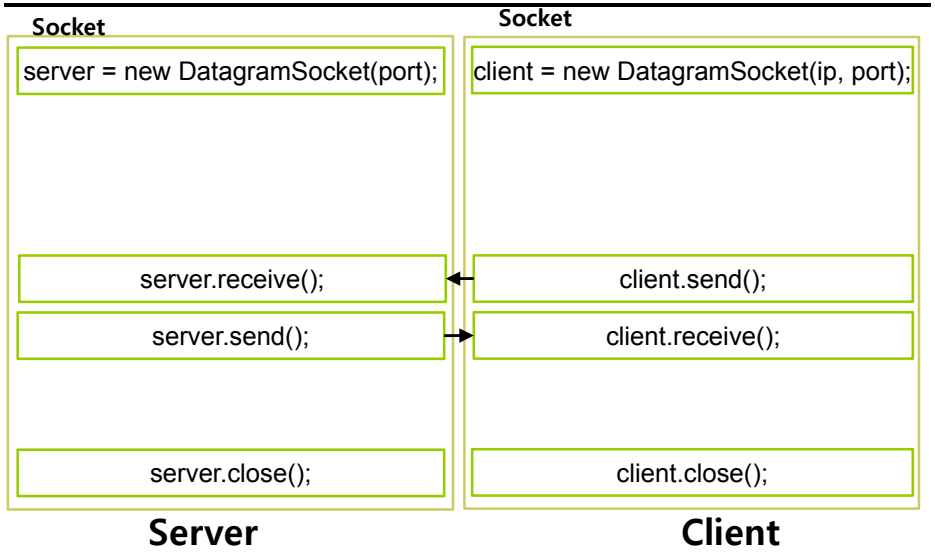
TCP Java Socket



UDP Socket

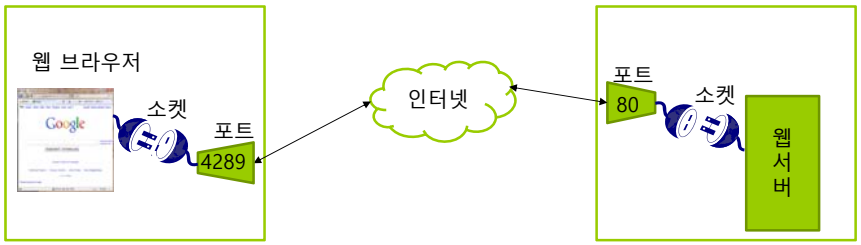


UDP Java Socket



Socket

- Socket
  - 소켓은 네트워크 상에서 수행되는 두 프로그램 간의 양방향 통신 링크의 한쪽 끝 단을 의미
  - 소켓은 특정 포트 번호와 연결되어 있음
    - TCP에서 데이터를 보낼 응용프로그램을 식별할 수 있음.
  - 자바에서의 데이타 통신 시 소켓 사용
  - 소켓 종류
    - 서버 소켓과 클라이언트 소켓



Socket 클래스

- **Socket** 클래스
  - **클라이언트** 소켓에 사용되는 클래스
  - java.net 패키지에 포함
  - 주요 생성자

생성자	설명
Socket(InetAddress address, int port)	소켓을 생성하여 지정된 IP 주소와 포트 번호에 연결한다.
Socket(String host, int port)	소켓을 생성하여 지정된 호스트와 포트 번호에 연결한다. 호스트 이름이 null인 경우는 루프백(loopback) 주소로 가정한다.

Socket 클래스

메소드	설명
void close()	소켓을 닫는다.
void connect(SocketAddress endpoint)	소켓을 서버에 연결
InetAddress getInetAddress()	소켓이 연결한 서버의 주소 반환
InputStream getInputStream()	소켓에 대한 입력 스트림 반환
InetAddress getLocalAddress()	소켓이 연결된 로컬 주소 반환
int getLocalPort()	소켓이 연결된 로컬 포트 번호 반환
int getPort()	소켓이 연결한 서버의 포트 번호 반환
OutputStream getOutputStream()	소켓에 대한 출력 스트림 반환
boolean isBound()	소켓이 로컬 주소에 연결되어있으면 true 반환
boolean isConnected()	소켓이 서버에 연결되어 있으면 true 반환
boolean isClosed()	소켓이 닫혀있으면 true 반환
void setSoTimeout(int timeout)	데이터 읽기 타임아웃 시간 지정. 0이면 타임아웃 해제.

소켓 생성, 서버 접속, 입출력 스트림 생성

- 클라이언트 소켓 생성 및 서버에 접속

```
Socket clientSocket = new Socket("128.12.1.1", 5550);
```

  - Socket 객체의 생성되면 곧 바로 128.12.1.1의 주소로 자동 접속
- 네트워크 입출력 스트림 생성

```
BufferedReader in = new BufferedReader(
                        new InputStreamReader(clientSocket.getInputStream()));
BufferedWriter out = new BufferedWriter(
                        new OutputStreamWriter(clientSocket.getOutputStream()));
```

  - 일반 스트림을 입출력 하는 방식과 동일
- 서버로 데이터 전송
  - flush()를 호출하면 스트림 속에 데이터를 남기지 않고 모두 전송

```
out.write("hello"+"₩n");
out.flush();
```

- 서버로부터 데이터 수신

```
int x = in.read();          // 서버로부터 한 개의 문자 수신
String line = in.readline(); //서버로부터 한 행의 문자열 수신
```

- 네트워크 접속 종료

```
clientSocket.close();
```

ServerSocket 클래스

- **ServerSocket** 클래스
  - **서버 소켓**에 사용되는 클래스
  - java.net 패키지에 포함
  - 주요 생성자

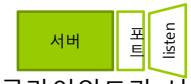
생성자	설명
ServerSocket(int port)	소켓을 생성하여 지정된 포트 번호에 연결한다.

  - 주요 메소드

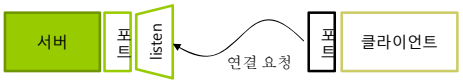
메소드	설명
Socket accept()	연결 요청을 기다리다 연결 요청이 들어오면 수락하고 새 Socket 객체를 반환
void close()	서버 소켓을 닫는다.
InetAddress getInetAddress()	서버 소켓에 연결된 로컬 주소 반환
int getLocalPort()	서버 소켓이 연결 요청을 모니터링하는 포트 번호 반환
boolean isBound()	서버 소켓이 로컬 주소에 연결되어있으면 true 반환
boolean isClosed()	서버 소켓이 닫혀있으면 true 반환
void setSoTimeout(int timeout)	accept()에 대한 타임 아웃 시간 지정. 0이면 타임아웃이 해제.

클라이언트와 서버 연결 순서

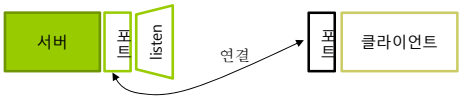
- 클라이언트와 서버 연결
  - 서버는 서버 소켓으로 들어오는 연결 요청을 기다림



  - 클라이언트가 서버에게 연결 요청



  - 서버가 연결 요청 수락하고 새로운 소켓을 만들어 클라이언트와 연결 생성



소켓 생성, 클라이언트 접속, 입출력 스트림 생성

- 서버 소켓 생성

```
ServerSocket serverSocket = new ServerSocket(5550);
```

  - 이미 사용 중인 포트 번호를 지정하면 오류가 발생
- 클라이언트로부터 접속 기다림

```
Socket socket = serverSocket.accept();
```

  - accept() 메소드는 연결 요청이 오면 새로운 Socket 객체 반환
  - 서버에서 클라이언트와의 데이터 통신은 새로 만들어진 Socket 객체를 통해서 이루어짐
  - ServerSocket 클래스는 Socket 클래스와 달리 주어진 연결에 대해 입출력 스트림을 만들어주는 메소드가 없음
- 네트워크 입출력 스트림 생성

```
BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
BufferedWriter out = new BufferedWriter(new
OutputStreamWriter(socket.getOutputStream()));
```

  - accept() 메소드에서 얻은 Socket 객체의 getInputStream()과 getOutputStream() 메소드를 이용하여 데이터 스트림 생성
  - 일반 스트림을 입출력하는 방식과 동일하게 네트워크 데이터 입출력

클라이언트로 데이터 송수신

- 클라이언트로부터 데이터 수신

```
int x = in.read();                // 클라이언트로부터 한 개의 문자 수신
String line = in.readline(); //클라이언트로부터 한 행의 문자열 수신
```

- 클라이언트로 데이터 전송

```
out.write("Hi!, Client"+"₩n");
out.flush();
```

  - flush()를 호출하면 스트림 속에 데이터를 남기지 말고 모두 전송
- 네트워크 접속 종료

```
socket.close();
```

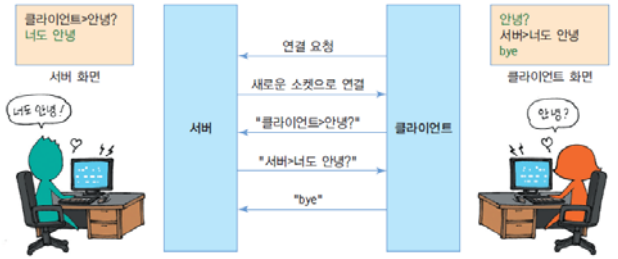
- 서버 응용프로그램 종료

```
serverSocket.close();
```

  - 더 이상 클라이언트의 접속을 받지 않고 서버 응용 프로그램을 종료하고자 하는 경우 ServerSocket 종료

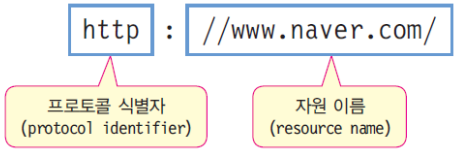
TCP/UDP 이용한 Server/Client 예제

- 간단한 채팅 프로그램 예제
  - 서버와 클라이언트가 1:1로 채팅 하는 간단한 예제
  - 서버와 클라이언트 간의 메시지 구분을 위해 서버는 메시지 앞에 "서버>"을 접두어로 붙여 메시지를 전송하며 클라이언트는 "클라이언트>"를 접두어로 붙여 메시지 전송
  - 서버와 클라이언트가 번갈아 가면서 메시지 전송 및 수신
  - 클라이언트가 bye를 보내면 프로그램 종료

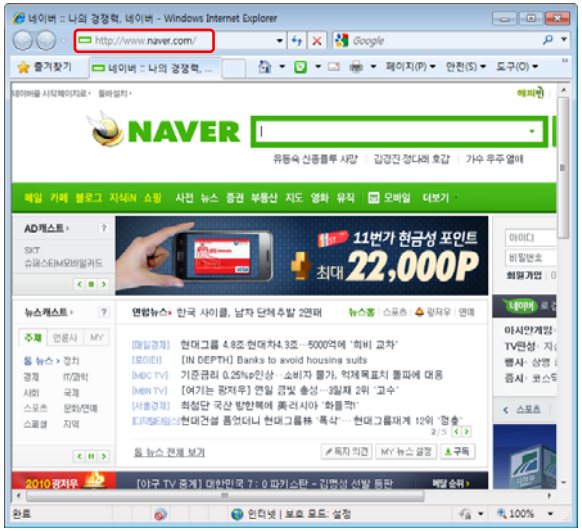


URL을 이용한 웹 프로그래밍

- URL이란?
  - URL은 Uniform Resource Locator
  - 인터넷 상의 리소스에 대한 주소
- URL 구조

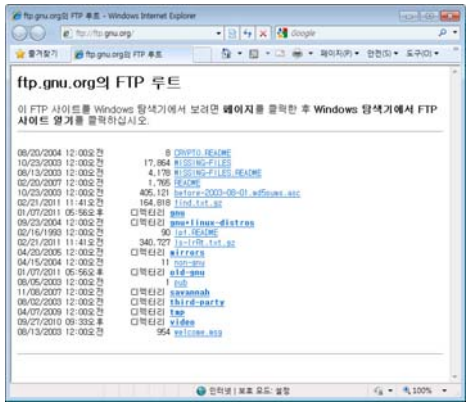


웹 브라우저 주소창의 URL



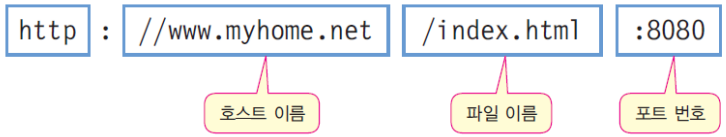
프로토콜 식별자

- 프로토콜(Protocol) 식별자
  - 인터넷상의 자원을 가져올 때 사용되는 통신 프로토콜 이름
  - 종류
    - HTTP, FTP, TELNET
  - 대부분의 브라우저들은 HTTP 외 다른 프로토콜도 지원



자원 이름

- 자원 이름
  - 자원 이름은 사용되는 프로토콜에 따라서 그 구성이 달라짐



HTTP의 경우

자바의 URL 클래스

- URL 클래스
  - java.net 패키지에 포함
  - 웹 상의 자원을 지정하는 URL을 나타냄

생성자	설명
URL(String spec)	문자열이 지정하는 자원에 대한 URL 객체를 생성
URL(String protocol, String host, int port, String file)	프로토콜 식별자, 호스트 주소, 포트 번호, 파일 이름이 지정하는 자원에 대한 URL 객체 생성
URL(String protocol, String host, String file)	프로토콜 식별자, 호스트 주소, 파일 이름이 지정하는 자원에 대한 URL 객체 생성
URL(URL context, String spec)	URL 객체 context에 대한 상대 경로가 지정하는 자원에 대한 URL 객체 생성

자바의 URL 클래스

메소드	설명
Object getContent()	URL의 컨텐트를 반환
String getFile()	URL 주소의 파일 이름 반환
String getHost()	URL 주소의 호스트 이름 반환
String getPath()	URL 주소의 경로 부분 반환
int getPort()	URL 주소의 포트 번호 반환
int getLocalPort()	소켓이 연결된 로컬 포트 번호 반환
int getPort()	소켓이 연결한 서버의 포트 번호 반환
InputStream openStream()	URL에 대해 연결을 설정하고 이 연결로부터 입력을 받을 수 있는 InputStream 객체 반환
URLConnection openConnection()	URL 주소의 원격 객체에 접속한 뒤 통신할 수 있는 URLConnection 객체 리턴

URL 객체 생성 방법

- 절대 경로로 URL 객체 생성

```
URL homePage = new URL("http://news.hankooki.com");
```

- 상대 경로로 URL 객체 생성

```
URL opinion = new URL(homePage, "opinion/editorial.htm");
```

opinion이라는 URL은 결국 다음 주소 의미
"http://news.hankooki.com/opinion/editorial.htm"

  - 잘못된 주소의 URL을 입력하면 MalformedURLException 예외 발생

예제: URL 파싱하기

URL 클래스를 이용하여 URL을 구성하는 프로토콜 이름, 호스트 주소, 포트 번호 등 각 부분을 파싱해보자

```
import java.net.*;
public class ParseURL {
    public static void main(String[] args) {
      URL opinion = null;
      URL homePage = null;
      try {
         homePage = new URL("http://news.hankooki.com:80"); // 절대 경로로 URL 객체 생성
          opinion = new URL(homePage,"opinion/editorial.htm"); // 상대 경로로 URL 객체 생성
      } catch (MalformedURLException e) {
         System.out.println("잘못된 URL입니다.");
      }
        System.out.println("protocol = " + opinion.getProtocol()); // 프로토콜 출력
        System.out.println("host = " + opinion.getHost()); // 호스트  이름 출력
        System.out.println("port = " + opinion.getPort()); // 포트 번호 출력
        System.out.println("path = " + opinion.getPath()); // 경로 부분 출력
        System.out.println("filename = " + opinion.getFile()); // 파일 이름 출력
    }
}
```

```
protocol = http
host = news.hankooki.com
port = 80
path = /opinion/editorial.htm
filename = /opinion/editorial.htm
```

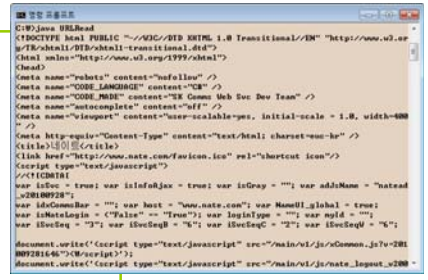
URL 객체를 이용하여 상대편으로부터 데이터 읽기

- URL 객체에서 데이터 읽기
  - URL 객체가 가리키는 주소에서 데이터를 가져올 때는 openStream() 메소드로 스트림 생성
  - openStream()이 리턴하는 InputStream 객체를 이용하여 일반 스트림 입력을 수행

예제: URL 주소에서 데이터 읽기

http 프로토콜로 www.nate.com 사이트에 접속한 뒤 www.nate.com에서 보내주는 웹 페이지를 받아보자.

```
import java.net.*;
import java.io.*;
public class URLRead {
    public static void main(String[] args) {
        try {
            // URL 객체 생성
            URL aURL = new URL("http://www.nate.com");
            // URL 객체에서 입력 스트림 생성
            BufferedReader in =
                  new BufferedReader(new InputStreamReader(
                                    aURL.openStream()));
            String inputLine;
            while ((inputLine = in.readLine()) != null) // 한행 씩 읽음
               System.out.println(inputLine);
            in.close();
        } catch (IOException e) {
            System.out.println("URL에서 데이터를 읽는 중 오류가
발생했습니다.");
        }
    }
}
```



URLConnection 클래스

- URLConnection 클래스
  - 주어진 원격지의 주소 URL에 네트워크 접속 후 데이터를 보내거나 받을 수 있도록 하는 기능
  - URL 객체 생성 방법
    - URL.openConnection() 이용

```
URL aURL = new URL("http://www.naver.com");
URLConnection uc = aURL.openConnection(); // 원격지와 연결한다.
```

    - URLConnection 생성자 이용

```
URL aURL = new URL("http://www.naver.com");
URLConnection uc = new URLConnection(aURL);
uc.connect(); // 원격지와 연결한다.
```

      - 연결하기 전에 여러 가지 인자들과 요청과 관련된 속성들을 설정 가능

URLConnection 클래스 주요 메소드

메소드	설명
abstract void connect()	URL에 의해 참조되는 외부 리소스와 통신 연결 설정
Object getContent()	URL 연결에서 컨텐트를 가져옴
String getContentEncoding()	컨텐트 인코딩 필드를 반환
int getContentLength()	컨텐트 길이 필드 반환
String getContentType()	컨텐트 타입 필드 반환
boolean getDoInput()	URLConnection 객체의 doInput 필드 값 반환
boolean getDoOutput()	URLConnection 객체의 doOutput 필드 값 반환
InputStream getInputStream()	설정된 연결에서 데이터를 읽을 입력 스트림 반환
OutputStream getOutputStream()	설정된 연결로 데이터를 출력할 출력 스트림 반환
URL getURL()	URLConnection 객체의 URL 필드 값 반환
void setDoInput(boolean doInput)	URLConnection 객체의 doInput 필드 값 설정
void setDoOutput(boolean doOutput)	URLConnection 객체의 doOutput 필드 값 설정

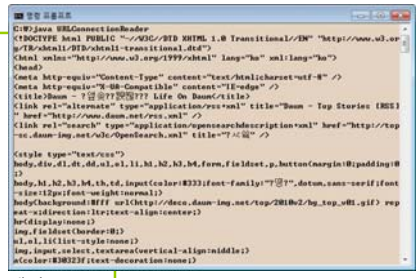
doInput 필드가 true로 설정되면 URLConnection 객체로 표현되는 URL 연결이 입력을 위해 사용됨을 의미. doOutput 필드가 true로 설정되면 출력을 위해 사용됨을 의미

URLConnection 객체를 이용하여 원격지 데이터 받기

- URLConnection 객체에서 데이터 읽기
  - URLConnection 객체에서 getInputStream() 메소드를 이용하여 입력 스트림을 얻은 후에 스트림 입력을 수행

예제 : URLConnection으로 원격지에서 데이터 읽기

URLConnection 객체를 이용하여 www.daum.net에 연결하여 데이터를 읽고 화면에 출력하는 프로그램을 작성하라.



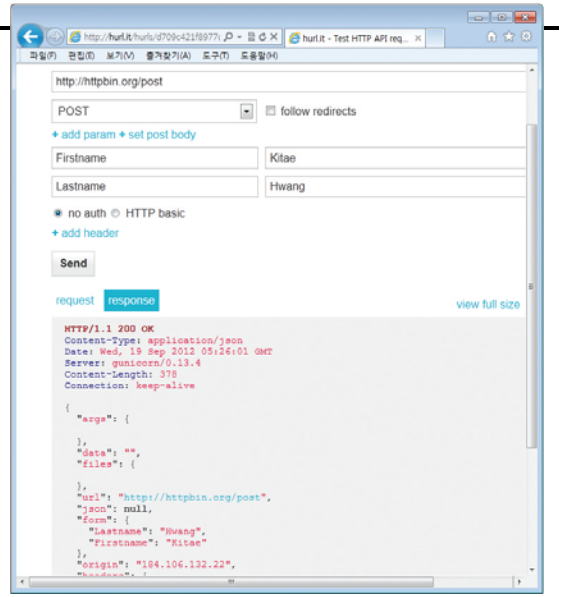
```
import java.io.*;
import java.net.*;
public class URLConnectionReader {
   public static void main(String[] args) {
      try {
         // URL 객체 생성
         URL aURL = new URL("http://www.daum.net");
         // URL 객체에서 URLConnection 객체 생성
         URLConnection uc = aURL.openConnection();
         BufferedReader in =
         new BufferedReader(new InputStreamReader(
                  uc.getInputStream())); // 입력 스트림 생성
         String inputLine;
            while ((inputLine = in.readLine()) != null) // 한행 씩 읽음
               System.out.println(inputLine);
               in.close();
      } catch (IOException e) {
         System.out.println("URL에서 데이터를 읽는 중 오류가
발생했습니다.");
      }
   }
}
```

URLConnection 객체를 이용하여 원격지로 데이터 보내기

- URLConnection 객체에서 데이터 쓰기
  - 웹 서버에 데이터를 요청하여 읽어올 때 주로 HTTP GET 방법 사용
  - 웹 서버에 데이터를 요청할 때 같이 처리될 데이터를 보낼 때 HTTP POST
    - HTTP POST를 이용하면 서버에 폼 (form) 데이터나 파일을 업로드할 수 있음
    - 요청과 같이 보내진 데이터를 서버가 처리하여 응답을 다시 클라이언트에 보냄
  - URLConnection 객체는 HTTP POST 방식으로 서버에 데이터 전송

HTTP POST사례

- 성과 이름을 입력하는 필드가 폼(form)
- 필드에 데이터를 입력한 후 "go" 버튼을 누르면 웹 서버로 데이터를 전송해야 하는데 이 데이터를 보내는 방법이 HTTP POST



서버에 데이터를 보내기 위한 단계

- 자바 프로그램이 웹 서버에 데이터를 보내기 위해서 필요한 단계
  1. URL 생성
  2. URL 객체에서 URLConnection 객체를 얻어온다.
  3. setDoOutput() 메소드로 doOutput 필드를 true로 설정
  4. connect() 메소드로 연결 설정
  5. 연결에서 출력 스트림을 얻어 온다.
  6. 출력 스트림에 데이터를 출력.
  7. 출력 스트림을 닫는다.

예제: URLConnection을 이용하여 웹 서버에 데이터 보내기

URLConnection 객체를 이용하여 웹 서버에 데이터를 보내고 웹 서버로부터 응답 데이터를 받아 화면에 출력하는 응용프로그램을 작성하라.

```
import java.io.*;
import java.net.*;
public class URLConnectionWriter {
   public static void main(String[] args) {
      try {
         // POST가 가능한 사이트 URL 객체 생성
         URL aURL = new URL("http://www.snee.com/xml/crud/posttest.cgi");
         URLConnection uc = aURL.openConnection(); // URLConnection 객체 생성
         uc.setDoOutput(true); // 출력 모드 설정
         OutputStreamWriter out = new OutputStreamWriter(uc.getOutputStream()); // 출력 스트림 생성
         out.write("fname=Kitae&lname=Hwang"); // 서버에 데이터 보내기
         out.close();
         BufferedReader in = new BufferedReader(
               new InputStreamReader(uc.getInputStream())); // 입력 스트림 생성
         String inputLine;
         while ((inputLine = in.readLine()) != null) // 한행 씩 읽음
            System.out.println(inputLine);
         in.close();
      } catch (IOException e) {
            System.out.println("URL에 데이터를 입출력 중에 오류가 발생했습니다.");
      }
   }
}
```