

제15장

자바 네트워크

구디아카데미 ▷ 민경태 강사

학습목표

1. 네트워크의 기본 개념에 대해서 알 수 있다.
2. 자바 네트워크 API에 대해서 알 수 있다.

```
each: function(e, t, n) {  
    i = 0,  
    o = e.length,  
    a = M(e);  
    if (n) {  
        if (a) {  
            for (; o > i; i++)  
                if (r = t.apply(e[i], n), r ===  
        } else  
            for (i in e)  
                if (r = t.apply(e[i], n), r ===  
    } else if (a) {  
        for (; o > i; i++)  
            if (r = t.call(e[i], i, e[i]))  
        } else  
            for (i in e)  
                if (r = t.call(e[i], i, e[i]))  
    return e  
},  
trim: b && !b.call("\uffff\u00a0") ?  
    return null == e ? "" : b.call(  
} : function(e) {  
    return null == e ? "" : (e + "  
},  
makeArray: function(e, t) {  
    var n = t || [];  
    return null != e && (M(Ob  
},  
isArray: function(e, t, n) {  
    var r;  
    if (t) {  
        if (n) return m.c  
        for (n = t.length  
            if (n in t  
    }  
}
```

목차

1. 네트워크의 기본 개념

- 1) 서버와 클라이언트
- 2) IP 주소
- 3) 도메인과 DNS
- 4) 포트
- 5) HTTP와 HTTPS
- 6) URI와 URL
- 7) OSI 7계층과 TCP 4계층

```
each: function(e, t, n) {  
    o = e.length,  
    a = M(e);  
    if (n) {  
        if (a) {  
            for (; o > i; i++)  
                if (r = t.apply(e[i], n), r ===  
        } else  
            for (i in e)  
                if (r = t.apply(e[i], n), r ===  
    } else if (a) {  
        for (; o > i; i++)  
            if (r = t.call(e[i], i, e[i]))  
    } else  
        for (i in e)  
            if (r = t.call(e[i], i, e[i]))  
    return e  
},  
trim: b && !b.call("\uffff\u00a0") ?  
    return null == e ? "" : b.call(  
} : function(e) {  
    return null == e ? "" : (e + "  
},  
makeArray: function(e, t) {  
    var n = t || [];  
    return null != e && (M(Obj  
},  
isArray: function(e, t, n) {  
    var r;  
    if (t) {  
        if (m) return m.c  
        for (n = t.length;  
            if (n in t)  
    }  
}
```

목차

2. 자바 네트워크 API

- 1) InetAddress 클래스
- 2) URL 클래스
- 3) HttpURLConnection 클래스
- 4) URLEncoder 클래스
- 5) URLDecoder 클래스

```

each: function(e, t, n) {
  var r, i = 0,
      o = e.length,
      a = M(e);
  if (n) {
    if (a) {
      for (; o > i; i++)
        if (r = t.apply(e[i], n), r === !1) break;
    } else
      for (i in e)
        if (r = t.apply(e[i], n), r === !1) break;
  } else if (a) {
    for (; o > i; i++)
      if (r = t.call(e[i], e[i]), r === !1) break;
  } else
    for (i in e)
      if (r = t.call(e[i], e[i]), r === !1) break;
  return e;
},
trim: b && !b.call("\uffff\u00a0") ? function(e) {
  return null == e ? "" : b.call(e);
} : function(e) {
  return null == e ? "" : (e + "").replace(C, "");
},
makeArray: function(e, t) {
  var n = t || [];
  return null != e && (M(Object(e)) ? x.merge(n, "string"
),
isArray: function(e, t, n) {
  var r;
  if (t) {
    if (n) return m.call(t, e, n);
    for (r = t.length, r = r ? 0 > n ? Math.max(0, r + n) : n; n in t && t[n] === e) return n;
  }
}

```

1. 네트워크의 기본 개념

네트워크

■ 네트워크

- 데이터를 주고 받을 수 있는 컴퓨터들과 주변 장치들의 집합

■ 노드

- 네트워크에 연결된 모든 장치들

■ 호스트

- 노드 중에서 애플리케이션을 실행할 수 있는 컴퓨팅 시스템을 갖춘 장치(PC, 노트북, 스마트폰 등)



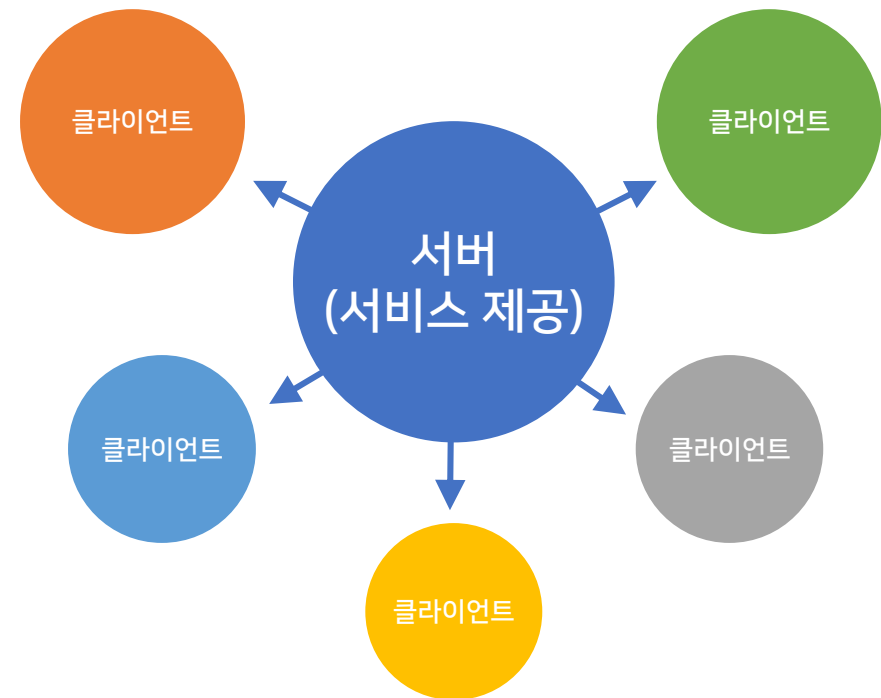
서버와 클라이언트

■ 서버(Server)

- 네트워크에서 데이터나 서비스를 제공하는 컴퓨터
- 제공하는 서비스에 따라서 웹 서버, 파일 서버, DB 서버 등으로 구분할 수 있음

■ 클라이언트(Client)

- 서버의 서비스를 이용하는 컴퓨터



IPv4

- IP version 4
- 32비트 숫자로 구성된 주소 체계
- 8비트 숫자(10진수로 표기, 0-255) 4개와 마침표(.)로 주소를 구성함
 - 0.0.0.0 ~ 255.255.255.255 사이 조합이 가능함
- IP 주소를 이용하여 모든 호스트를 구분하므로 모든 호스트는 고유의 IP 주소를 가져야 함
- 현재 IPv4는 포화 상태로 이를 해결하기 위한 IPv6가 존재함

IPv6

- IP version 6
- IPv4의 주소 고갈 문제를 해결하기 위해 나타남
- 128비트 숫자로 구성된 주소 체계
- 16비트 숫자 8개(16진수로 표기)와 콜론(:)으로 주소를 구성함
 - 예시) FE80:6E78:E1D2:C35D:FECD:BA98:7652:EABF

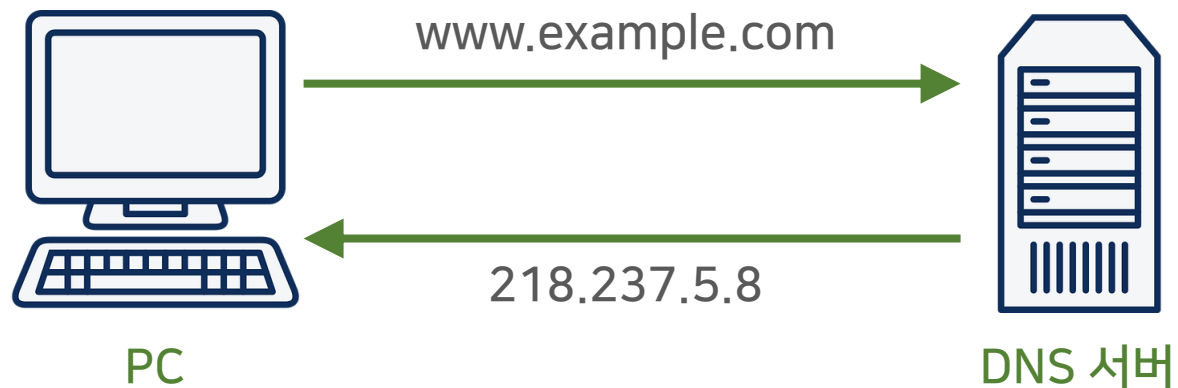
도메인과 DNS

■ 도메인

- 외우고 다니기 어려운 IP 주소 대신 사용하는 네트워크 호스트 이름
- www.example.com처럼 문자로 구성되어 있음

■ DNS

- Domain Name System
- 도메인을 IP 주소로 바꿔서 호스트를 찾아갈 수 있게 하는 시스템



포트

- 논리적인 통신 연결 번호
- 특정 프로세스나 서비스에 연결하기 위한 연결 번호
- 컴퓨터에서 서로 다른 종류의 트래픽을 구분하기 위한 연결 번호
- 포트번호는 임의로 사용할 수는 있지만 예약된 번호를 사용하게 되면 통신에 문제가 될 수 있음
- 0-65,535 사이 번호를 가짐

포트 구분

- IANA(Internet Assigned Numbers Authority, 인터넷 할당 번호 관리기관)에서 구분한 포트의 사용 영역은 3가지로 나뉜다
- 포트 번호 사용 영역
 1. 0-1,023
 - 잘 알려진 포트 (well known port)
 - 예약된 영역
 - root 권한(관리자 권한)으로 bind 할 수 있음
 2. 1,024-49,151
 - 등록된 포트 (registered port)
 - 일반 유저 권한으로 bind 할 수 있음
 3. 49,152-65,535
 - 동적 포트 (dynamic port)
 - 서버-클라이언트 통신 시 자동으로 bind 되는 영역

Well Known Ports

- 잘 알려진 포트로 0~1,023번 사이에 배정된 포트
- 주요 Well Known Ports

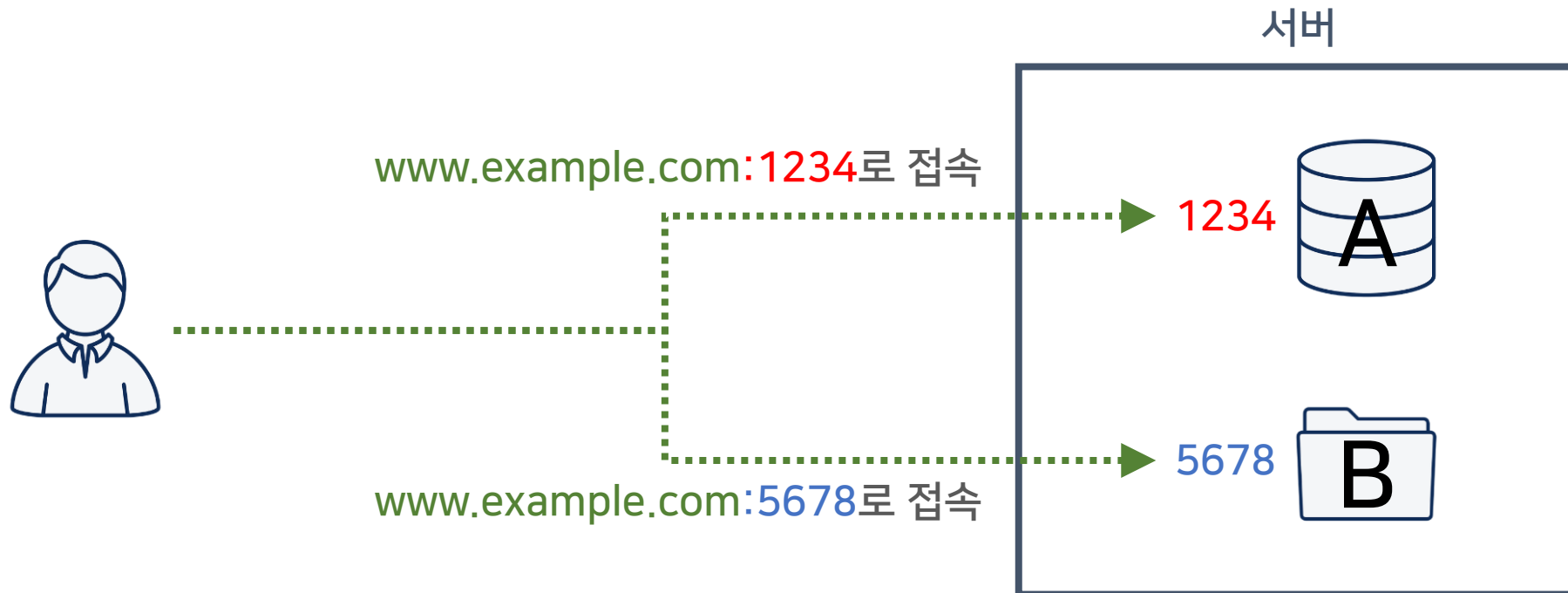
포트	서비스	의미
21	FTP, File Transfer Protocol	파일 전송
22	SSH, Secure SHell	보안 원격 컴퓨터 접속
23	Telnet	일반 원격 컴퓨터 접속
25	SMTP, Simple Mail Transfer Protocol	이메일 보내기
80	HTTP, HyperText Transfer Protocol	하이퍼텍스트 전송 (웹 페이지 전송)
110	POP3, Post Office Protocol version 3	이메일 받기
443	HTTPS, HTTP Secure	SSL 기반의 하이퍼텍스트 전송 (보안)

포트가 필요한 이유

- 하나의 서버가 가진 IP 주소는 하나임
- 하지만 하나의 서버가 여러 개의 서비스를 제공할 수 있음
- IP 주소만으로는 각 서비스를 구분할 수 없음
- 각 서비스마다 고유의 포트 번호를 할당함
- 클라이언트는 서버의 IP 주소와 이용하려는 서비스의 포트 번호를 이용하여 각 서비스를 구분해서 요청할 수 있음

포트 활용

- 호스트 주소와 포트는 콜론(:)으로 연결함
 - 호스트 주소 ▶ `www.example.com`
 - A 서비스 포트 ▶ `1234`
 - B 서비스 포트 ▶ `5678`




HTTP와 HTTPS

■ HTTP

- Hyper Text Transfer Protocol
- 하이퍼텍스트 전송 프로토콜
- 웹 문서인 HTML 문서를 전송하는 프로토콜
- 암호화되지 않는 평문으로 데이터를 전송함

■ HTTPS

- HTTP Secure
- 보안 처리된 HTTP
- SSL(Secure Sockets Layer)을 이용해서 보안 처리함
- SSL 인증서 발급에는 소정이 비용이 소요됨
- 암호화된 데이터를 전송하기 때문에 보안이 우수함

 이 사이트는 보안 연결(HTTPS)이 사용되지 않았습니다.
이 사이트에 입력하는 비밀번호나 신용카드 번호 등의 정보가 공격자에 의해 도용될 수 있습니다. [자세히 알아보기](#)

HTTP 사용시 경고 메시지

URI와 URL

■ URI

- Uniform Resource Identifier
- 통합 자원 식별자
- 어떤 자원을 식별할 수 있는 고유의 문자열을 의미함

■ URL

- Uniform Resource Locator
- 통합 자원 위치 정보
- 어떤 자원이 어디 있는지를 표시한 문자열로 URI의 일종임

URL 구성 요소

■ URL의 구성 요소

- protocol
- host
- port
- path
- query
- reference

The diagram illustrates the components of the URL `https://www.example.com:9090/board/view.html?no=1#footnote`. The URL is segmented into the following parts:

- protocol (schema):** `https`
- host:** `www.example.com`
- port:** `9090`
- path:** `/board/view.html`
 - file:** `view.html`
 - extension:** `.html`
- query:** `?no=1`
- reference (fragment):** `#footnote`

OSI 7계층

- 통신 과정을 구성 요소와 역할에 따라 7단계로 정의한 국제 통신 표준 규약

7계층	응용 계층	응용 프로그램 간의 통신 지원 및 데이터 교환 서비스 지원
6계층	표현 계층	데이터 변환(인코딩, 디코딩, 압축, 암호화)
5계층	세션 계층	응용 프로그램 간의 연결 유지 및 지원
4계층	전송 계층	데이터 전송 준비(큰 데이터를 작은 조각으로 나눔)
3계층	네트워크 계층	데이터가 목적 시스템으로 전송될 수 있도록 주소 추가
2계층	데이터 링크 계층	물리적 매체에 패킷을 전송하기 위한 프레임 생성
1계층	물리 계층	통신 케이블을 이용해 전기 신호를 비트 스트림으로 전송

TCP/IP 프로토콜

- Transmission Control Protocol/Internet Protocol
- 노드 간에 통신할 수 있도록 만든 프로토콜 (통신규약) 모음
- 인터넷의 기반이 되는 네트워크 시스템
- 하드웨어, 운영체제, 노드 종류에 상관 없이 동작하는 개방형 구조
- 다수의 프로토콜 중에 가장 중요한 TCP와 IP로 이름을 붙임

TCP/IP 4계층

■ TCP/IP 4계층 구조도



OSI 7계층과 TCP/IP 4계층

■ OSI 7계층과 TCP/IP 4계층 비교

OSI 7계층	프로토콜	TCP/IP 4계층
응용 계층	HTTP, FTP, TELNET DNS, SMTP, POP3 ...	응용 계층
표현 계층		
세션 계층		
전송 계층	TCP, UDP	전송 계층
네트워크 계층	IPv4, IPv6	인터넷 계층
데이터 링크 계층	Ethernet, 802.11, PPP	네트워크 접근 계층
물리 계층	케이블, 커넥터, 신호 규격	

```

each: function(e, t, n) {
  var r, i = 0,
      o = e.length,
      a = M(e);
  if (n) {
    if (a) {
      for (; o > i; i++)
        if (r = t.apply(e[i], n), r === !1) break
    } else
      for (i in e)
        if (r = t.apply(e[i], n), r === !1) break
  } else if (a) {
    for (; o > i; i++)
      if (r = t.call(e[i], e[i]), r === !1) break
  } else
    for (i in e)
      if (r = t.call(e[i], e[i]), r === !1) break;
  return e
},
trim: b && !b.call("\uffff\u00a0") ? function(e) {
  return null == e ? "" : b.call(e)
} : function(e) {
  return null == e ? "" : (e + "").replace(C, "")
},
makeArray: function(e, t) {
  var n = t || [];
  return null != e && (M(Object(e)) ? x.merge(n, "string"
),
isArray: function(e, t, n) {
  var r;
  if (t) {
    if (n) return m.call(t, e, n);
    for (r = t.length, r = r ? 0 > n ? Math.max(0, r + n) :
      if (n in t && t[n] === e) return n
  }
}

```

2. 자바 네트워크 API

java.net 패키지

- 네트워크 애플리케이션 구현을 위한 클래스를 제공하는 패키지
- 네트워크 리소스에 보다 쉽게 접근하기 위한 High Level 추상화 클래스를 제공함
 - InetAddress 클래스
 - URI 클래스
 - URL 클래스
 - URLConnection 클래스
 - HttpURLConnection 클래스

InetAddress 클래스

- IP 주소와 호스트 이름을 관리하는 클래스
- public 생성자를 지원하지 않고 객체 생성을 위한 static 메소드를 지원함
- 자바 프로그램을 작성할 때 IP 주소가 필요한 경우에는 InetAddress 클래스를 활용함

InetAddress 클래스 - 객체 생성

■ InetAddress 객체 생성을 위한 정적 메소드

메소드	의미
<code>static InetAddress[] getAllByName(String host)</code> <code>throws UnknownHostException</code>	String host에 대응하는 InetAddress[] 배열 반환
<code>static InetAddress getByAddress(byte[] addr)</code> <code>throws UnknownHostException</code>	byte[] addr에 대응하는 InetAddress 객체 반환
<code>static InetAddress getByAddress(String host, byte[] addr)</code> <code>throws UnknownHostException</code>	String host와 byte[] addr에 대응하는 InetAddress 객체 반환
<code>static InetAddress getName(String host)</code> <code>throws UnknownHostException</code>	String host에 대응하는 InetAddress 객체 반환
<code>static InetAddress getLocalHost()</code> <code>throws UnknownHostException</code>	localhost의 InetAddress 객체 반환

InetAddress 클래스 - 메소드

■ InetAddress 클래스 주요 메소드

메소드	의미
byte[] getAddress()	InetAddress 객체의 실제 IP 주소를 byte[] 배열로 반환
String getHostAddress()	IP 주소를 문자열로 반환
String getHostName()	IP 주소에 대응하는 호스트 이름을 문자열로 반환
String toString()	"호스트/IP 주소" 형식의 문자열을 반환

InetAddress 클래스 활용

- InetAddress 클래스를 이용한 호스트 정보 확인하기

```
public class MainClass {  
    public static void main(String[] args) {  
        try {  
            InetAddress iAddr1 = InetAddress.getByName("www.google.com");  
            System.out.println(iAddr1.getHostAddress());  
            System.out.println(iAddr1.getHostName());  
            byte[] b = {(byte)172, (byte)217, (byte)161, (byte)228};  
            InetAddress iAddr2 = InetAddress.getByAddress(b);  
            System.out.println(iAddr2.getHostAddress());  
            System.out.println(iAddr2.getHostName());  
        } catch (UnknownHostException e) {  
            e.printStackTrace();  
        }  
    }  
}
```

실행결과 (다른 결과가 나올 수 있음)

"172.217.161.228"
"www.google.com"

"172.217.161.228"
"kix06s05-in-f4.1e100.net"

URL 클래스

- World Wide Web의 리소스(자원)에 대한 포인터를 나타냄
- 파일/디렉터리처럼 단순한 리소스나 데이터베이스/검색엔진에 대한 쿼리처럼 복잡한 리소스 모두 가능함
- 잘못된 URL을 이용해 URL 객체를 생성하면 `MalformedURLException` 예외가 발생함

URL 클래스 - 생성자

■ URL 클래스 주요 생성자

생성자	의미
URL(String spec) throws MalformedURLException	String URL 구문을 이용해 URL 객체를 생성
URL(String protocol, String host, String file) throws MalformedURLException	String 프로토콜, 호스트, 파일을 이용해 URL 객체를 생성
URL(String protocol, String host, int port, String file) throws MalformedURLException	String 프로토콜, 호스트, 파일과 int 포트번호를 이용해 URL 객체를 생성

URL 클래스 - 메소드

■ URL 클래스 주요 메소드

메소드	역할
String getProtocol()	URL의 프로토콜을 반환
String getHost()	URL의 호스트를 반환
String getPort()	URL의 포트번호를 반환
String getPath()	URL의 경로를 반환
String getQuery()	URL의 쿼리를 반환
String getFile()	URL의 파일(경로 + 쿼리)을 반환
URLConnection openConnection() throws IOException	URL이 참조하는 원격 개체에 대한 연결을 나타내는 URLConnection 객체를 반환
final InputStream openStream() throws IOException	URL에 대한 연결을 열고 해당 연결을 읽기 위한 InputStream 객체를 반환

URL 클래스 활용

■ URL 클래스를 이용한 정보 확인하기

```
public class MainClass {  
    public static void main(String[] args) {  
        try {  
            URL url = new URL("https://www.example.com:9090/board/view.html?no=1");  
            System.out.println(url.getFile());  
            System.out.println(url.getHost());  
            System.out.println(url.getPath());  
            System.out.println(url.getPort());  
            System.out.println(url.getProtocol());  
            System.out.println(url.getQuery());  
        } catch (MalformedURLException e) {  
            e.printStackTrace();  
        }  
    }  
}
```

실행결과

```
"/board/view.html?no=1"  
"www.example.com"  
"/board/view.html"  
9090  
"https"  
"no=1"
```


URLConnection 클래스

- 응용 프로그램과 URL 간의 통신 링크를 나타내는 모든 클래스들의 슈퍼 클래스
- URL에서 참조하는 리소스를 읽고 쓸 수 있는 클래스
- URL 객체로부터 URLConnection을 얻을 수 있음

URLConnection 클래스 - 메소드

■ URLConnection 클래스 주요 메소드

메소드	역할
<code>void addRequestProperty(String key, String value)</code>	요청 속성을 key-value 형식으로 등록
<code>Map<String, List<String>> getRequestProperties()</code>	모든 요청 속성을 Map<String, List<String>> 형식으로 반환
<code>String getRequestProperty(String key)</code>	요청 속성 중 key 값을 가진 value 반환
<code>abstract void connect()</code>	URL이 참조하는 리소스에 대한 통신 링크 열기
<code>InputStream getInputStream()</code>	open connection으로부터 InputStream 객체 얻기
<code>OutputStream getOutputStream()</code>	open connection으로부터 OutputStream 객체 얻기
<code>void setDoOutput(Boolean dooutput)</code>	dooutput이 true이면 애플리케이션이 URL 연결에 데이터를 쓸 수 있음

URLConnection 클래스

- HTTP 관련 기능을 지원하는 URLConnection 클래스의 서브 클래스
- 각종 응답 코드(Response Code)를 필드 값으로 가지고 있어 응답 여부에 따른 코드 작성이 가능함
- 주요 응답 코드

응답 코드	의미	필드
200	정상 응답	URLConnection.HTTP_OK
404	페이지 없음	URLConnection.HTTP_NOT_FOUND
500	내부 서버 오류	URLConnection.HTTP_INTERNAL_ERROR

URLConnection 클래스 활용

- HttpURLConnection 클래스를 활용해 연결된 URL로부터 접속정보를 확인하거나 수정할 수 있음
- 접속 정보로부터 InputStream을 얻어낸 뒤 데이터를 읽어들이 수 있음
- 접속 정보로부터 Reader를 얻어낼 수는 없으므로 텍스트 정보를 읽어야 하는 경우에는 InputStream을 얻어낸 뒤 이를 InputStreamReader로 변환해서 사용해야 함

URLConnection 클래스 - 메소드

- URLConnection 클래스의 서브 클래스이므로 URLConnection 클래스의 메소드를 모두 사용할 수 있음
- HttpURLConnection 클래스 주요 메소드

메소드	역할
abstract void disconnect()	연결 닫기. 곧 서버에 대한 다른 요청이 없음을 의미함 다른 요청을 만들기 위해서 disconnect() 메소드를 호출하면 안 됨
InputStream getErrorStram()	connection이 실패하였으나 서버가 데이터를 보냈다면 해당 데이터를 읽을 수 있는 오류 스트림 반환
String getRequestMethod()	요청 메소드 반환
int getResponseCode()	응답 코드 반환
void setRequestMethod(String method)	요청 메소드 설정 (GET, POST, HEAD, OPTIONS, PUT, DELETE, TRACE 중 선택)

접속 정보 확인

- HttpURLConnection 클래스를 이용한 접속 정보 확인하기

```
public class MainClass {  
    public static void main(String[] args) {  
        try {  
            URL url = new URL("https://www.google.com/");  
            HttpURLConnection con = (HttpURLConnection)url.openConnection();  
            System.out.println(con.getResponseCode());  
            System.out.println(con.getRequestMethod());  
            System.out.println(con.getRequestProperty("User-Agent"));  
            con.disconnect();  
        } catch (IOException e) {  
            e.printStackTrace();  
        }  
    }  
}
```

실행결과


```
200  
"GET"  
"Java/17.0.8.1"
```

이진 데이터 가져오기

- HttpURLConnection 클래스를 이용한 이미지 파일 가져오기

```
public class MainClass {  
    public static void main(String[] args) {  
        try {  
            String spec = "https://www.gdu.co.kr/images/main/logo.png";  
            URL url = new URL(spec);  
            HttpURLConnection con = (HttpURLConnection)url.openConnection();  
            BufferedInputStream in = new BufferedInputStream(con.getInputStream());  
            File file = new File(spec.substring(spec.lastIndexOf("/") + 1));  
            BufferedOutputStream out = new BufferedOutputStream(new FileOutputStream(file));  
            byte[] b = new byte[1024];  
            int readByte = 0;  
            while((readByte = in.read(b)) != -1)  
                out.write(b, 0, readByte);  
            out.close();  
            in.close();  
        } catch(IOException e) {  
            e.printStackTrace();  
        }  
    }  
}
```

실행결과

 logo.png
PNG 파일
6.10KB

텍스트 데이터 가져오기

- HttpURLConnection 클래스를 이용한 텍스트 데이터 읽기

```
public class MainClass {  
    public static void main(String[] args) {  
        try {  
            URL url = new URL("https://www.gdu.co.kr/");  
            HttpURLConnection con = (HttpURLConnection)url.openConnection();  
            BufferedReader br = new BufferedReader(new InputStreamReader(con.getInputStream()));  
            String line = null;  
            StringBuilder sb = new StringBuilder();  
            while((line = br.readLine()) != null) {  
                sb.append(line).append("\n");  
            }  
            sb.deleteCharAt(sb.length() - 1);  
            System.out.println(sb.toString());  
            br.close();  
        } catch(IOException e) {  
            e.printStackTrace();  
        }  
    }  
}
```

실행결과

```
"<!DOCTYPE html>"  
"..."  
"</html>"
```


URL Encoding과 URL Decoding

■ URL Encoding

- 웹 상에서 문자열을 안전하게 전송할 수 있는 형식으로 바꾸는 것을 의미함
- "application/x-www-form-urlencoded" 형식을 따름
- 권장하는 인코딩 방식은 UTF-8 방식임

■ URL Decoding

- URL Encoding된 문자를 다시 원래의 문자로 되돌리는 것을 의미함



URLEncoder 클래스

- URL Encoding을 위한 자바 유틸리티 클래스
- 인코딩 방식
 - "A-Z", "a-z", "0-9" 그대로 사용
 - 특수문자 ".", "-", "*", "_" 그대로 사용
 - 공백 " "은 "+" 기호로 변환
 - 다른 모든 문자는 바이트로 변환한 뒤 3자리 문자열 "%xy" 방식으로 변환
(xy는 변환된 바이트의 16진수 표현법)

URLEncoder 클래스 - 메소드

- URLEncoder 객체 생성 없이 곧바로 사용 가능한 static 메소드가 존재하고 있음

메소드	의미
static String encode(String s, String enc) throws UnsupportedOperationException	String s를 명시된 String enc 방식으로 인코딩한 문자열을 반환
static String encode(String s, Charset charset)	String s를 명시된 Charset charset 문자셋으로 인코딩한 문자열을 반환

URLDecoder 클래스

- 인코딩된 데이터를 다시 원래의 값으로 되돌리는 디코딩을 위한 자바 유틸리티 클래스
- URLEncoder 클래스를 이용해 인코딩한 데이터를 다시 되돌릴 때 사용함

URLDecoder 클래스 - 메소드

- URLDecoder 객체 생성 없이 곧바로 사용 가능한 static 메소드가 존재하고 있음

메소드	의미
<code>static String decode(String s, String enc)</code> <code>throws</code> <code>UnsupportedEncodingException</code>	String s를 명시된 String enc 방식으로 디코딩한 문자열을 반환
<code>static String decode(String s, Charset charset)</code>	String s를 명시된 Charset charset 문자셋으로 디코딩한 문자열을 반환

URLEncoder/URLDecoder 클래스 예시

■ URLEncoder/URLDecoder 클래스 활용하기

```
public class MainClass {  
    public static void main(String[] args) {  
        try {  
            String originalData = "홍길동 abc ABC 123 .-_*?!@";  
            System.out.println("원본: " + originalData);  
            String encodeData = URLEncoder.encode(originalData, "UTF-8");  
            System.out.println("암호화: " + encodeData);  
            String decodeData = URLDecoder.decode(encodeData, "UTF-8");  
            System.out.println("복호화: " + decodeData);  
        } catch (UnsupportedEncodingException e) {  
            e.printStackTrace();  
        }  
    }  
}
```

실행결과

"원본: 홍길동 abc ABC 123 .-_*?!@"

"암호화: %ED%99%8D%EA%B8%B8%EB%8F%99+abc+ABC+123+.-_*%3F%21%40"

"복호화: 홍길동 abc ABC 123 .-_*?!@"