16장 네트워크 프로그래밍

네트워크의 개념

◈ 네트워크(Network)

데이터 교환을 목적으로 로컬 컴퓨터와 원격 컴퓨터 간 데이터 흐름을 나타내는 구조

◈ IP(Internet Protocol)

인터넷 주소라 불리는 32비트 숫자로 구성된 주소체계로 구성되며, 인터넷에 연결되어 있는 장치를 식별할 수 있도록 각각의 장치에 부여되는 고유 주소이다.

◈ 포트(Port)

각각의 프로그램에 부여되는 주소라고 보면 된다.

IP를 통해 하나의 컴퓨터를 찾은 후 해당 컴퓨터에 실행되고 있는 여러 개의 서버 프로그램 중 하나에 접속하기 위해 해당 프로그램의 포트를 같이 사용한다.

포트번호는 0~65535까지 지정할 수 있으며, 0~1023은 시스템상 예약된 포트번호로 가급적 사용하지 않는다.

네트워크의 개념

◈ 프로토콜(Protocol)

클라이언트와 서버간의 통신 규약

상호간의 접속이나 절단방식, 통신방식, 데이터의 형식, 전송속도 등에 대해 정의하는 것

◈ 클라이언트(Client)

서버에 요청을 보내고 데이터를 받아오는 컴퓨터를 말한다.

◈ 서버(Server)

클라이언트로부터 요청이 오면, 요청에 맞는 데이터를 전달할 준비가 되어 있는 컴퓨터를 말한다.

네트워크의 개념

◈ TCP/IP(Transmission Control Protocol/Internet Protocol)

인터넷상에서 호스트들을 서로 연결시키는데 사용하는 통신 프로토콜로 기종이 서로 다른 컴퓨터 시스템을 서로 연결해 데이터를 전송하기 위한 통신 프로토콜

◈ UDP(User Datagram Protocol)

IP를 사용하는 네트워크 내에서 컴퓨터들 간에 메시지들을 교환할 때 제한된 서비스만을 제공하는 통신 프로토콜

◈ URL(Uniform Resource Locator)

인터넷상에서 접근 가능한 자원(Resource)의 주소를 나타내는 표준

◈ URI(Uniform Resource Identifier)

특정 자원에 접근하기 위한 형식이나 고유한 이름으로 URL보다 넓은 의미의 개념이다.

네트워크의 개념

◈ Broadcast

데이터를 여러 방향으로 동시에 전송하여 동일 IP그룹에 있는 컴퓨터라면 데이터를 수신할 수 있는 방식

◈ Unicast

특정한 대상 수신자에게만 데이터를 보내는 방식

◈ Multicast

다중의 수신 대상자들에게 데이터를 보내는 방식

◈ RMI(Remote Method Invocation)

자바 프로그래밍 언어와 개발 환경을 사용하여 서로 다른 컴퓨터 상에 있는 객체들이 분산 네트워크 내에서 상호 작용하는 객체지향형 프로그램을 작성할 수 있도록 해주는 방식이다. RMI는 일반적으로 RPC라고 알려진 것의 자바 버전이다.

네트워크의 개념

◈ Broadcast

데이터를 여러 방향으로 동시에 전송하여 동일 IP그룹에 있는 컴퓨터라면 데이터를 수신할 수 있는 방식

◈ Unicast

특정한 대상 수신자에게만 데이터를 보내는 방식

◈ Multicast

다중의 수신 대상자들에게 데이터를 보내는 방식

◈ RMI(Remote Method Invocation)

자바 프로그래밍 언어와 개발 환경을 사용하여 서로 다른 컴퓨터 상에 있는 객체들이 분산 네트워크 내에서 상호 작용하는 객체지향형 프로그램을 작성할 수 있도록 해주는 방식이다. RMI는 일반적으로 RPC라고 알려진 것의 자바 버전이다.

TCP 통신을 위한 클래스들

◈ InetAddress

IP 주소를 표현한 클래스

반환형	메서드	설명
InetAddress[]	getAllByName(String host)	매개변수 host에 대응되는 InetAddress 배열을 반환한다.
InetAddress	getByAddress(byte[] addr)	매개변수 addr에 대응되는 InetAddress 객체를 반환한다.
	getByAddress(String host, byte[] addr)	매개변수 host와 addr로 Inetaddress객체를 생성한다.
	getByName(String host)	매개변수 host에 대응되는 InetAddress 객체를 반환한다.
	getLocalHost()	로컬호스트의 InetAddress 객체를 반환한다.

반환형	메서드	설명
byte[]	getAddress()	InetAddress 객체의 실제 IP 주소를 바이트 배열로 리턴한다.
String	getHostAddress()	IP 주소를 문자열로 반환한다.
	getHostName()	호스트 이름을 문자열로 반환한다.
	toString()	IP 주소를 스트링 문자열로 오버라이딩 한 메소드

TCP 통신을 위한 클래스들

◈ URL 클래스

URL을 추상화하여 만든 클래스

```
<protocol>://<host>:<port>/<path>?<query>#<reference>
http://www.daum.net:80/member/mem.jsp?name=sung#content
```

◈ URLConnection 클래스

원격자원에 접근하는데 필요한 정보를 갖고 있다.

원격서버의 헤더 정보, 해당 자원의 길이와 타입정보, 언어 등을 얻어 올 수 있다.

```
URL  url  =  new URL("http://java.sun.com");
URLConnection  urlCon  =  url.openConnection();
urlCon.connect();
```

TCP 통신을 위한 클래스들

◈ Socket

서버 프로그램으로 연결 요청을 하며, 데이터 전송을 담당한다.

[표 15-7] Socket 클래스의 주요 생성자

생성자	설명
Socket(InetAddress address, int port)	InetAddress 객체와 port를 이용하여 Socket 객체를 생성한다.
Socket(String host , int port)	host와 port를 이용하여 Socket 객체를 생성한다.

TCP 통신을 위한 클래스들

◈ Socket

[표 15-8] Socket 클래스의 주요 메서드

반환형	메서드	설명
void	close( )	소켓 객체를 닫는다.
InetAddress	getInetAddress( )	소켓 객체를 InetAddress 객체로 반환한다.
InputStream	getInputStream( )	소켓 객체로부터 입력할 수 있는 InputStream 객체를 반환한다.
InetAddress	getLocalAddress( )	소켓 객체의 로컬 주소를 반환한다.
int	getPort( )	소켓 객체의 포트를 반환한다.
boolean	isClosed( )	소켓 객체가 닫혀있으면 true를, 열려있으면 false를 반환한다.
	isConnected( )	소켓 객체가 연결되어 있으면 true, 연결되어 있지 않으면 false를 반환한다.
void	setSoTimeout(int timeout)	소켓 객체의 시간을 밀리 세컨드로 설정한다.

TCP 통신을 위한 클래스들

◈ ServerSocket

서버 프로그램에서 사용하는 소켓으로, 포트를 통해 연결 요청이 오기를 대기한다.

요청이 오면 클라이언트와 연결을 맺고 또 다른 소켓을 만드는 역할을 한다.

이때 새로 만들어진 소켓이 클라이언트의 소켓과 데이터를 주고 받는다.

[표 15-9] ServerSocket 클래스의 주요 생성자

생성자	설명
ServerSocket(int port)	port를 이용하여 ServerSocket 객체를 생성한다.

TCP 통신을 위한 클래스들

◈ ServerSocket

[표 15-10] ServerSocket 클래스의 주요 메서드

반환형	메서드	설명
Socket	accept( )	클라이언트의 Socket 객체가 생성될 때까지 블로킹되는 메서드다. 클라이언트의 Socket 객체가 생성되면 서버에서 클라이언트와 통신할 수 있는 Socket 객체를 반환하게 된다.
void	close( )	ServerSocket 객체를 닫는다.
int	getLocalPort( )	ServerSocket 객체가 청취하고 있는 포트번호를 반환한다.
	getSoTimeout( )	ServerSocket 클래스의 accept( ) 메서드가 유효할 수 있는 시간을 밀리 세컨드로 반환한다. 만약, 0이면 무한대를 의미한다.
boolean	isClosed( )	ServerSocket 객체의 닫힌 상태를 반환한다.
void	setSoTimeout(int timeout)	ServerSocket 클래스의 accept( ) 메서드가 유효할 수 있는 시간을 밀리 세컨드로 설정해야 한다. 만약, 시간이 지나면 java.net.SocketTimeoutException 예외가 발생하는데, 이 예외가 발생하더라도 ServerSocket 객체는 계속 유효하다.

TCP 통신을 위한 클래스들

◈ 클라이언트와 서버 간 통신