

오픈 소스 라이브러리

1. 오픈 소스

❖오픈 소스: 소프트웨어 혹은 하드웨어 제작자의 권리를 지키면서 원시 코드를 누구나 열람할 수 있도록 한 소프트웨어 혹은 오픈 소스 라이선스에 준하는 모든 것

❖오픈 소스의 장점

- ① 편리성: 사용자가 직접 처음부터 개발하려고 하면 그 만큼의 시간과 노동력이 소요됨
- ② 뛰어난 성능: 대부분 대수의 개발자들이 협업해서 만들었기 때문에 성능이 우수할 가능성이 높음
- ③ 신뢰성: 오픈 소스 라이브러리의 버전이 높다면 많은 프로그래머들이 지속적으로 기능과 성능을 검증해 주었으므로 신뢰성이 높다.

2. Java 오픈 소스

❖ 카산드라 데이터베이스(Cassandra Database)

- ▷ 관련 URL : <http://cassandra.apache.org/>
- ▷ 확장성을 위해서 '분산 저장 시스템 기능'을 제공하는 동시에 RDBMS와 좀 더 간단한 데이터 저장 형식을 지원
- ▷ 기존의 RDBMS에서 사용하던 테이블 대신 Hashtable과 같이 Key/Value 형식의 데이터 저장 형식을 갖고 있음 → 'NoSQL 데이터베이스'

❖ 하둡(Hadoop)

- ▷ 관련 URL : <http://hadoop.apache.org/>
- ▷ 아파치 재단에서 제공하는 오픈 소스 분산 처리 시스템으로서 매우 획기적인 방법으로 분산 처리 플랫폼을 제공

❖ 루씬(Lucene)

- ▷ 관련 URL : <http://lucene.apache.org/>
- ▷ 고성능의 인덱싱(색인 작업) 및 텍스트 기반의 검색 엔진 플랫폼

2. Java 오픈 소스

❖ 메이븐(Maven)

- ▷ 관련 URL : <http://maven.apache.org/>
- ▷ Java Build 및 외부 라이브러리 사용을 편리하게 하는 도구

❖ 아파치 웹 서버(Apache WEB Server)

- ▷ 관련 URL : <http://httpd.apache.org/>
- ▷ 윈도우 계열부터 유닉스 계열의 서버까지 다양한 플랫폼에서 동작하고 기능 확장성이 뛰어난 웹 서버

❖ Log4j

- ▷ Java 클래스에서 로그를 생성 및 기록하고 관리하기 위한 오픈 소스 프레임 워크

2. Java 오픈 소스

❖ Commons Project

❖ 관련 URL : <http://commons.apache.org/>

컴포넌트	설명
BeanUtils	클래스가 어떤 메소드와 속성으로 구성되어 있는지 알 수 있는 기능을 제공하는 리플렉션(reflection)과 객체의 상태와 그 정보를 수집하고 변경할 수 있는 인트로스펙션(introspection)을 쉽게 구현할 수 있는 API를 제공한다.
Collections	앞서 알아본 ArrayList, Hashmap과 같은 자바 기본 컬렉션(Collections)과 관련된 기능의 확장이라고 생각하면 된다.
DBCP	자바 프로그램에서 데이터베이스로부터 데이터를 가져오기 위해서는 Connection이라는 객체가 필요하다. 이 Connection 객체를 생성하려면 시스템 리소스가 많이 필요하며, IO 관련 클래스와 같이 반환(close)을 해주어야 한다. 이렇게 Connection 객체들을 미리 만들어 놓고 저장한 다음 애플리케이션이 데이터베이스에 접속할 때마다 저장한 Connection 객체를 전달하는 기능을 풀링(pooling)이라고 한다. 풀링은 또한 Connection 객체의 개수를 제한하는 등 다양한 제어 기능을 제공하고 있다. DBCP 컴포넌트는 이런 풀링 기능을 제공해준다.

2. Java 오픈 소스

Email	이메일을 발송하는데 필요한 여러 가지 기능들을 모아 놓은 컴포넌트다.
FileUpload	일반적으로 웹 환경의 클라이언트에서 서버로 파일을 전송하는데 사용하는 컴포넌트다.
HttpClient	웹에서 사용하는 HTTP 프로토콜을 사용해서 클라이언트가 할 수 있는 기능을 제공해준다. 예를 들어 HttpClient를 사용하여 특정 웹 사이트의 정보를 가져오거나 서버와 데이터를 주고 받을 수 있다.
JCS	자바 캐시 시스템(Java Cache System)으로 메모리에 데이터를 올려놓고 데이터를 전달해주는 기능을 한다. 보통 쓰기 빈도가 적고, 읽기 빈도가 많은 데이터들을 메모리와 같은 읽기 속도가 빠른 저장 매체에 올려놓기 위해서 사용한다. 읽기 속도가 빠르다는 장점이 있다.



CSV

CSV

- ❖ 항목을 쉼표(,)로 구분하여 나열한 표 형식의 텍스트 데이터를 CSV(Comma-Separated Values)라고 한다.
- ❖ CSV는 첫 번째 줄에 정의 항목의 명칭을 작성하고, 두 번째 행부터는 데이터를 작성한다. 각 항목은 쉼표로 구분되어 있으며, 항목 안에 쉼표가 있는 경우는 항목을 큰 따옴표(")로 둘러싼다.
- ❖ Microsoft Excel에서도 열 수 있으며 검색 및 편집이 용이하다.

CSV

❖ CSV 데이터는 항목이 쉼표로 구분되어 있기 때문에 자바에서 읽어 들일 때 '쉼표를 구분자로 split 메서드를 실행하면 된다'고 생각하기 쉽다. 그러나 실제로는 그렇게 간단하지가 않다.

❖ 예를 들어 항목 자체에 쉼표가 포함될 수도 있기 때문에 단순히 쉼표를 구분자로 split 메서드를 실행했을 경우 하나의 항목을 2개로 분할할 가능성도 있다. 그 경우 큰따옴표 기호로 문자열을 둘러싸는 것이 일반적인 대처 방안이지만 그러면 이번에는 문자열 안에 큰따옴표가 들어 있을 때의 대처가 필요하다. 이것만으로도 복잡하지만 그 외에도 다음과 같은 사항을 생각하면 필요한 프로그램의 양은 점점 늘어간다.

- ✓ 첫 번째 행은 헤더 행으로 하고 싶다
- ✓ · 들어 있는 값이 올바른지 체크하고 싶다
- ✓ · 각 행의 내용을 자바의 객체에 각각 넣고 싶다
- ✓ · 쓰기도 하고 싶다

❖ CSV 처리가 쉽게 보이기도 하지만 실은 내용이 복잡하다. 그러한 고려하여 제대로 데이터를 읽어주는 라이브러리가 Super CSV다.

employee.csv

name,age,birth,email,note

배유나,35,1978/4/1,bae@xxx.co.kr,"소유 면허: 1종 운전면허, 응용 정보 기술자"

문정원,28,1985/10/23,moon@xxx.co.kr,



Employee.java

```
import java.util.Date;
public class Employee {
    private String name;
    private Integer age;
    private Date birth;
    private String email;
    private String note;
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public Integer getAge() {
        return age;
    }
    public void setAge(Integer age) {
        this.age = age;
    }
    public Date getBirth() {
        return birth;
    }
}
```

Employee.java

```
public void setBirth(Date birth) {
    this.birth = birth;
}
public String getEmail() {
    return email;
}
public void setEmail(String email) {
    this.email = email;
}
public String getNote() {
    return note;
}
public void setNote(String note) {
    this.note = note;
}
@Override
public String toString() {
    return "Employee [name=" + name + ", age=" + age + ",
    birth=" + birth + ", email=" + email + ", note=" + note
    + "]\n";
}
}
```

CSVMain.java

```
import java.nio.file.Files;  
import java.nio.file.Path;  
import java.nio.file.Paths;  
import java.util.ArrayList;  
import java.util.List;
```

```
import org.supercsv.cellprocessor.Optional;  
import org.supercsv.cellprocessor.ParseDate;  
import org.supercsv.cellprocessor.ParseInt;  
import org.supercsv.cellprocessor.constraint.NotNull;  
import org.supercsv.cellprocessor.constraint.StrRegex;  
import org.supercsv.cellprocessor.ift.CellProcessor;  
import org.supercsv.io.CsvBeanReader;  
import org.supercsv.io.ICsvBeanReader;  
import org.supercsv.prefs.CsvPreference;
```

CSVMain.java

```
public class CSVMain {  
  
    public static void main(String[] args) {  
        CellProcessor[] processors = new CellProcessor[] {  
            new NotNull(), // name  
            new ParseInt(new NotNull()), // age  
            new ParseDate("yyyy/MM/dd"), // birth  
            new StrRegEx("[a-z0-9¥¥._]+@[a-z0-9¥¥._]+"), // email  
            new Optional() // note  
        };  
        List<Employee> list = new ArrayList<Employee>();  
        Path path = Paths.get("./employee.csv");  
    }  
}
```

CSVMain.java

```
try (ICsvBeanReader beanReader = new
CsvBeanReader(Files.newBufferedReader(path),
                CsvPreference.STANDARD_PREFERENCE)) {
    String[] header = beanReader.getHeader(true);
    Employee employee;
    while ((employee = beanReader.read(Employee.class,
header, processors)) != null) {
        list.add(employee);
    }
    list.stream().forEach(System.out::println);
} catch (Exception e) {
    System.out.println(e.getMessage());
    e.printStackTrace();
}
```

CSVMain.java

```
path = Paths.get("employeecopy.csv");
String[] header = new String[] { "name", "age", "birth", "email",
    "note" };
CellProcessor[] writeprocessors = new CellProcessor[] { new
    NotNull(), // name
    new ParseInt(new NotNull()), // age
    new NotNull(), // birth
    new StrRegex("[a-z0-9¥¥._]+@[a-z0-9¥¥.]+"), // email
    new Optional() // note
    };
```


CSVMain.java

```
        try (ICsvBeanWriter beanWriter = new
CsvBeanWriter(Files.newBufferedWriter(path),
                CsvPreference.STANDARD_PREFERENCE)) {
            beanWriter.writeHeader(header);
            for (Employee employee : list) {
                beanWriter.write(employee, header,
writeprocessors);
            }
        } catch (Exception e) {
            System.out.println(e.getMessage());
            e.printStackTrace();
        }
    }
}
```

JSON Parsing

JSON 파싱

▷ JSON을 사용함으로써 얻을 수 있는 이점

① 데이터 크기가 작다

- ▷ XML에 사용되던 태그가 없기 때문에 전체 텍스트의 사이즈가 줄어들었음

② 사람이 읽고 쓰기에 용이하다

- ▷ 괄호와 세미콜론 같은 간단한 기호로 구성
- ▷ 데이터 형이 바이너리가 아니라 순수 텍스트(Plain text)

③ 개발 언어와 OS 종류에 구애받지 않는다

- ▷ JSON은 순수 텍스트로 구성되어 있고 데이터 교환을 위한 표준 표기 방법
- ▷ 이기종 시스템 간에 데이터 교환 방식

JSON 파싱

▶ JSON 객체

▷ JSON 규칙

- ▷ JSON 객체의 시작과 끝은 중괄호 ({ })로 표기한다
- ▷ JSON 객체의 속성은 Name/Value 형태다
- ▷ 속성 Name/Value는 콜론 (:)을 기준으로 구분한다

JSON 파싱

▶ JSON 객체

▷ JSON 데이터 표현 방법

```
사용예: {"NAME" : "Benjamin Kim"}  
        {"URL" : "www.facebook.com"}
```

▷ JSON 객체의 속성은 콤마를 사용해서 구분한다

```
사용예: { "NAME" : "Benjamin Kim", "URL" : "www.facebook.com" }
```

▷ JSON 객체의 속성 name은 객체 내에서 유일해야 된다

▷ String형 데이터는 쌍따옴표를 사용한다

```
사용예: case 1 : {"name" : "BenjaminKim"}  
        case 2 : {"age" : 33 }  
        case 3 : {"isMale" : true }  
        case 4 : {"name" : "Benjamin Kim" , "age" : 33 , "isMale" : true }
```

JSON 파싱

▶ JSON 객체

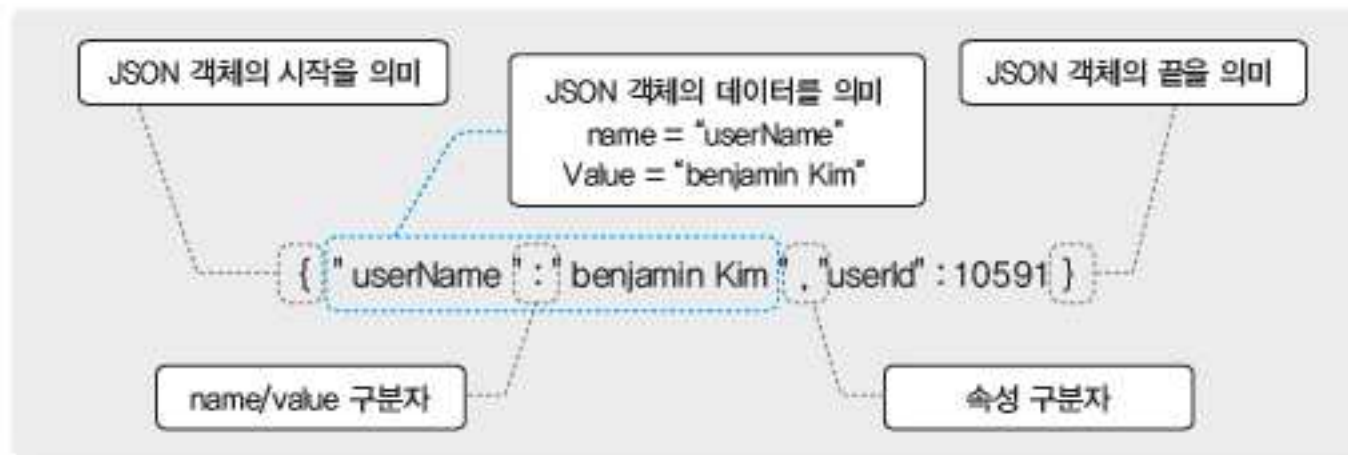
- ▶ JSON 문법에서 객체는 자바의 Hashtable 객체와 매우 비슷한 원리로 데이터를 저장
- ▶ 다음과 같이 Hashtable 객체를 생성하고 데이터를 화면에 출력한다고 가정

```
Hashtable userInfo = new Hashtable();  
userInfo.put("userName", "benjamin Kim");  
userInfo.put("userId", 10591);  
  
System.out.println(userInfo.get("userName"));  
System.out.println(userInfo.get("userId"));
```

- ▶ JSON을 사용한 데이터 표기법은 매우 간단하면서 XML보다 데이터 크기도 줄일 수 있는 것을 확인

JSON 파싱

▶ JSON 객체



△ 그림 20-2 Hashtable의 데이터를 json으로 표현한 데이터

JSON 파싱

▶ JSON 배열

▷ JSON 배열을 만드는 표기법

- ▷ JSON 배열의 시작과 끝은 대괄호를 사용
- ▷ JSON 배열의 멤버 변수들은 콤마를 사용해서 구분
- ▷ JSON 배열의 구성 아이템으로 JSON 객체도 넣을 수 있음
- ▷ JSON 객체의 value 부분에는 JSON 객체를 넣을 수 있음

사용예: ["Benjamin Kim", "Hamzani Ali", "Lodoss"]

[1, 2, 3, 5, 8, 13, 21]

[{ "name": "Benjamin" , "age" : 33 } , { "name" : "Hamzani" , "age" : 30 }]

JSON 파싱

JSON 클래스 라이브러리 다운로드 - www.mvnrepository.com에서 JSON 검색해서 다운로드



JSON 파싱

▶ json 라이브러리의 주요 클래스들

클래스	설명
org.json.simple Class JSONObject	JSON 객체를 추상화한 클래스로, java.util.HashMap 클래스를 상속받고 있으므로 대부분의 메소드가 HashMap 클래스로부터 상속받고 있다.
org.json.simple Class JSONArray	JSON 배열을 추상화한 클래스로, java.util.ArrayList 클래스를 상속하고 있으므로 메소드 사용 방법은 대부분 우리가 이미 배운 ArrayList와 거의 흡사하다.
org.json.simple.parser Class JSONParser	JSON 데이터를 파싱하는 기능을 구현한 클래스다.
org.json.simple Class JSONValue	JSON 데이터를 다루기 위한 몇 가지 메소드들을 제공한다.
org.json.simple.parser Class ParseException	JSONParser 클래스를 사용해서 파싱할 때 발생할 수 있는 예외 사항을 추상화한 클래스다.

배열

❖ JSON 배열을 받아서 파싱

```
package json;
```

```
import org.json.JSONArray;
```

```
public class JSONMain {
```

```
    public static void main(String[] args) {
```

```
        String json = "[100, 500, 300, 200, 400]";
```

```
        JSONArray ar = new JSONArray(json);
```

```
        for(int i=0; i<ar.length(); i++){
```

```
            System.out.println(ar.get(i));
```

```
        }
```

```
    }
```

```
}
```



배열

❖ JSON 객체를 받아서 파싱

```
package json;
```

```
import org.json.JSONObject;
```

```
public class JSONMain {
```

```
    public static void main(String[] args) {
```

```
        String json = "{color: ₩red₩,value: ₩#f00₩}";
```

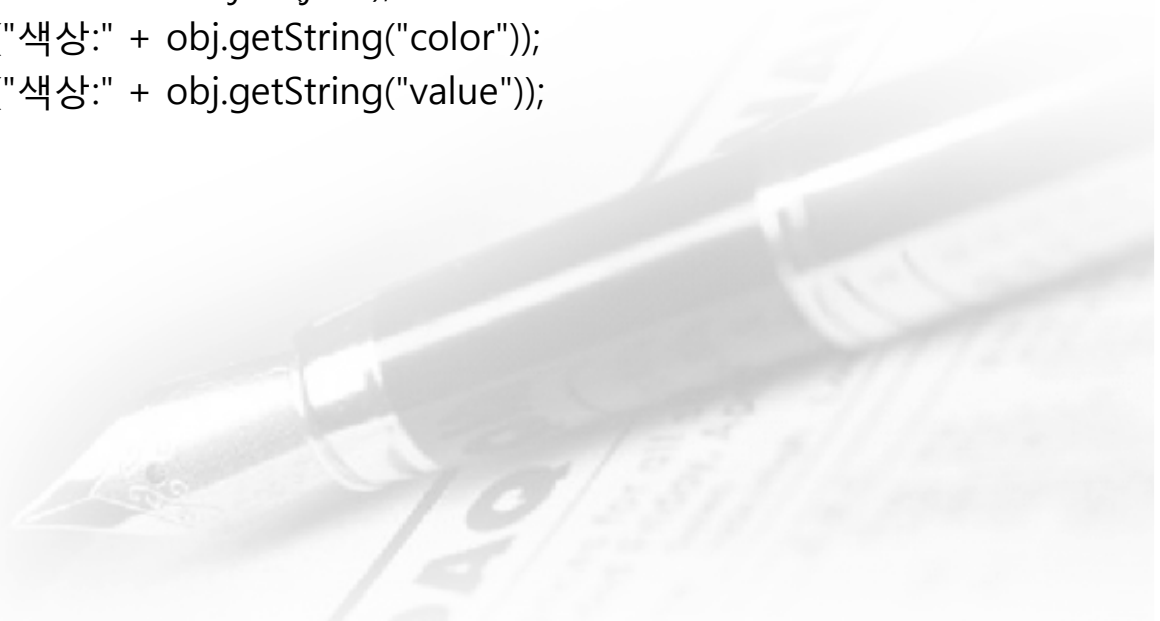
```
        JSONObject obj = new JSONObject(json);
```

```
        System.out.println("색상:" + obj.getString("color"));
```

```
        System.out.println("색상:" + obj.getString("value"));
```

```
    }
```

```
}
```



다음 오픈 API



도서정보 JSON

❖ 데이터를 다운로드 받아서 파싱을 수행 해 줄 클래스 (JsonThread.java)

```
class JsonThread extends Thread{
    public void run() {
        StringBuilder sBuffer = new StringBuilder();
        String json="";
        ArrayList<String> data = new ArrayList<String>();
        try {
            String urlAddr =
                "https://dapi.kakao.com/v3/search/book?target=title&query=java";
            URL url = new URL(urlAddr);
            HttpURLConnection conn = (HttpURLConnection)
                url.openConnection();
```



도서정보 JSON

```
if (conn != null) {
    conn.setConnectTimeout(20000);
    conn.setUseCaches(false);
    conn.addRequestProperty("Authorization", "KakaoAK 06fab290c9f4eb6f130c09796d57bc30");

    if (conn.getResponseCode() == HttpURLConnection.HTTP_OK) {
        InputStreamReader isr = new InputStreamReader(
            conn.getInputStream());
        BufferedReader br = new BufferedReader(isr);

        while (true) {
            String line = br.readLine();
            if (line == null) {
                break;
            }
            sBuffer.append(line);
        }
        br.close();
        conn.disconnect();
    }
}
```

도서정보 JSON

```
json = sBuffer.toString();

        System.out.println(json);
    } catch (Exception e) {
        System.out.println("가져오기 실패:" + e.getMessage());
    }
    try {
        JSONObject obj = new JSONObject(json);
        JSONArray documents = obj.getJSONArray("documents");
        for (int i = 0; i < documents.length(); i++) {
            JSONObject book = documents.getJSONObject(i);
            data.add("제목:" + book.getString("title") + " 가격:" +
book.getInt("price"));
        }
    }
    catch (Exception e) {
        System.out.println("파싱 실패:" + e.getMessage());
    }
    System.out.println(data);
}
}
```


도서정보 JSON

❖ 메인 클래스 (Main.java)

```
package network;  
public class Main {  
  
    public static void main(String[] args) {  
        JsonThread th = new JsonThread();  
        th.start();  
    }  
}
```



XML Parsing

XML 파싱

- ❖ **RSS(Really Simple Syndication, Rich Site Summary):** 자주 바뀌는 내용을 제공하기 위해 사용되는 표준 웹 피드 포맷으로 xml 형식을 주로 이용
 - XML은 웹 서비스의 기본 데이터 포맷으로 서버와 클라이언트의 통신 수단
 - HTML을 파싱하여 문서를 화면에 출력하는 방법은 상호 호환성에 문제
 - HTML로 문서의 구조나 정보를 표현하는 것에는 한계
 - 이런 단점을 극복하기 위해서 XML 표준 규격이 발표
 - 표준 규격은 다른 OS나 개발 언어 환경에서도 서로 공용(share)할 수 있다는 장점



XML 파싱

▶ XML이 사용되는 분야

- ▶ XML을 활용하면 구조화된 데이터를 저장한 문서를 만들 수 있으며
보다 자유로운 문서 표현이 가능
 - ① 데이터 저장과 표현
 - ▶ 트리(Tree) 형태로 구조화될 수 있어 프로그램에서 데이터를 읽고 쓰기에 매우 적합
 - ② 데이터 공유
 - ▶ 데이터가 문서에 저장되므로 어떤 프로그램이든지 쉽게 내용을 확인할 수 있기 때문에
개발 언어 혹은 OS에 매우 독립적인 형식
 - ③ 시스템/프로그램의 유연성 증대
 - ▶ 대부분의 오픈 소스 라이브러리나 프레임워크에서는 프로그램 외부에서
기능에 대한 정의 및 설정을 XML 파일로 조정할 수 있음
 - ④ 자유로운 문법
 - ▶ XML은 태그와 속성을 개발자가 자유롭게 선언할 수 있음

최소의 문법에 대해서만 정의하고 있으므로 데이터를 표현하는데 매우 자유로움

XML 파싱

- 서버는 클라이언트의 요청을 받아들여 처리하고 그 결과를 **XML**로 리턴하며 클라이언트는 **XML**을 분석하여 처리 결과를 얻습니다.
- **XML** 자체는 단순한 텍스트 포맷이지만 규칙이 엄격
- **XML** 파서는 크게 **DOM**, **SAX** 두 가지로 구분되며 **DOM**은 트리 형식으로 문서를 읽어서 전체 구조를 파악한 후 정보를 구하는 방식이고 **SAX**는 순차적으로 문서를 읽으면서 정보를 차례대로 읽는 방식입니다.
- **DOM**은 메모리를 많이 사용하지만 성능이 좋고 **SAX**는 느리지만 메모리를 거의 사용하지 않는다.



XML 파싱

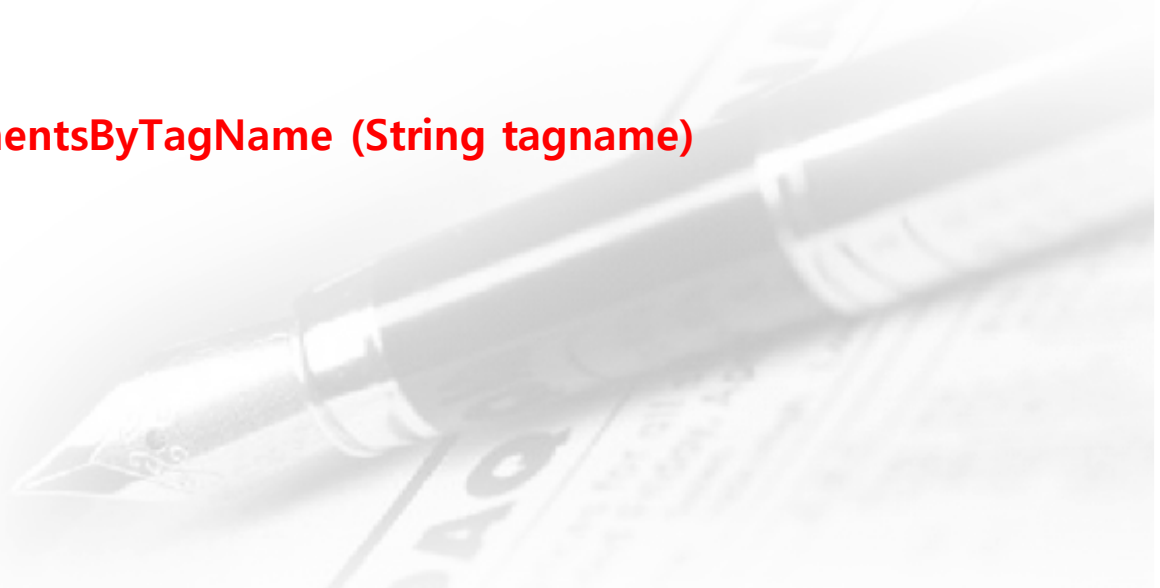
❖ DOM 파서

- XML 문서를 파싱하기 위해서는 **DocumentBuilderFactory** 객체를 생성해야 합니다.
- 이 클래스는 추상 클래스이므로 직접 생성할 수 없으며 **newInstance**라는 정적 메소드로 생성합니다.
- 객체를 생성한 후 주석 무시 여부, 네임 스페이스 인식 여부, 유효성 점검 여부 등의 속성을 설정합니다.
- 속성을 맞춘 후 **newDocumentBuilder** 메소드를 호출하면 **DocumentBuilder** 객체가 리턴됩니다.
 - **static DocumentBuilderFactory newInstance ()**
 - **DocumentBuilder newDocumentBuilder ()**
- **DocumentBuilder**도 추상 클래스이므로 반드시 이 과정을 거쳐야만 생성할 수 있습니다.
 - **Document parse (InputStream stream [, String systemId])**
 - **Document parse (String uri)**
 - **Document parse (File file)**

XML 파싱

❖ DOM 파서

- **Parse**메소드는 스트림을 분석하여 메모리에 트리 형태로 전개해 놓고, 이후 **Document** 객체의 메소드로 요소들을 빠른 속도로 읽을 수 있습니다.
 - **Element Document.getDocumentElement ()**
- **XML** 문서는 유일한 루트 엘리먼트 하나를 가지는데 **getDocumentElement** 메소드는 루트 엘리먼트를 구합니다.
- 다음 메소드는 태그명과 일치하는 엘리먼트를 찾아 **Node**의 배열인 **NodeList** 객체를 리턴합니다.
 - **NodeList Element.getElementsByTagName (String tagname)**



XML 파싱

❖ DOM 파서

- **NodeList**에는 개수를 구하는 **getLength** 메소드와 순서값으로부터 노드를 찾는 **item** 메소드가 제공된다.
- **Node** 객체를 구했으면 다음 메소드로 노드의 정보를 구한다.
 - **String getNodeName ()**
 - **short getNodeType ()**
 - **String getNodeValue ()**
- 각각 노드의 이름, 타입, 값이며, 다음 노드를 중심으로 주변 노드를 찾는 메소드다.
 - **Node getFirstChild ()**
 - **Node getLastChild ()**
 - **Node getNextSibling ()**
 - **Node getPreviousSibling ()**
 - **Node getParentNode ()**
 - **NodeList getChildNodes ()**

XML 파싱

❖ DOM 파서

- **Item** 엘리먼트의 값을 읽으려면 **getFirstChild**로 첫 번째 자식을 구하고 **getNodeValue**로 자식의 값을 읽어야 합니다.
- **DOM**은 엘리먼트 안의 문자열도 하나의 객체로 취급하며 문자열을 엘리먼트의 자식을 취급하기 때문입니다.
- 항목 여러 개를 읽어야 하므로 **NodeList**의 배열을 순회해야 하며, 엘리먼트의 속성을 읽을 때는 다음 메소드를 사용합니다.
 - **NamedNodeMap Node.getAttributes ()**
- 엘리먼트와는 달리 속성은 순서가 없으므로 어떤 속성이 먼저 조사될지는 알 수 없습니다.
- **XML** 스펙은 속성의 순서에 의미를 부여하지 않으며, **NamedNodeMap**은 순서가 없는 속성을 저장합니다.

기상청 날씨

<http://www.kma.go.kr/weather/forecast/mid-term-xml.jsp?stnId=109>

```
<?xml version="1.0" encoding="UTF-8"?>
<wid>
  <header>
    <title>서울, 경기도, 육상주간예보</title>
    <tm>201403110600</tm>
    <wf>
      <![CDATA[
        기압골의 영향으로 19일에 비가 오겠고, 그 밖의 날은 고기압의 가장자리에 들어 가끔 구름많겠습니다.<br />기온은
        />강수량은 평년(강수량 : 1~2mm)과 비슷하거나 조금 많겠습니다.
      ]]>
    </wf>
  </header>
  <body>
    <location city="11B10101" province="11B00000" wl_ver="3">
      <province code="11B00000">서울·인천·경기도</province>
      <city code="11B10101">서울</city>
      <data>
        <numEf>2</numEf>
        <tmEf>2014-03-13</tmEf>
        <wf>구름많음</wf>
        <tmn>3</tmn>
        <tmx>10</tmx>
        <reliability>보통</reliability>
      </data>
      <data>
        <numEf>3</numEf>
        <tmEf>2014-03-14</tmEf>
        <wf>구름많음</wf>
        <tmn>-2</tmn>
        <tmx>9</tmx>
        <reliability>보통</reliability>
      </data>
    </location>
  </body>
</wid>
```

[10, 9, 13, 15, 15, 14, 8, 7, 11, 13, 13, 12, 11, 10, 14, 16, 16, 15, 10, 8, 12, 14, 14, 13]


기상청 날씨

❖ 데이터를 받아와서 파싱하는 클래스(DomThread.java)

```
import java.io.*;
import java.net.*;
import java.util.*;
import javax.xml.parsers.*;
import org.w3c.dom.*;

public class DOMThread extends Thread {
    String xml;
    List<String> data = new ArrayList<String>();

    public void run() {
        StringBuilder sBuffer = new StringBuilder();
        try {
```



기상청 날씨

```
String urlAddr =  
    "http://www.kma.go.kr/weather/forecast/mid-term-  
xml.jsp?stnId=109";  
  
URL url = new URL(urlAddr);  
URLConnection conn =  
    (URLConnection)url.openConnection();  
if (conn != null) {  
    conn.setConnectTimeout(20000);  
    conn.setUseCaches(false);  
    if (conn.getResponseCode() ==  
        HttpURLConnection.HTTP_OK) {  
        InputStreamReader isr = new  
            BufferedReader(new InputStreamReader(conn.getInputStream()));  
        //줄 단위로 읽어서 sBuffer에 저장  
        while(true) {  
            String line = isr.readLine();  
            if (line == null) {  
                break;  
            }  
            sBuffer.append(line);  
        }  
    }  
}
```

기상청 날씨

```
        br.close();
        conn.disconnect();
    }
}
//전부 읽었으면 String으로 변환
xml = sBuffer.toString();
} catch (Exception e) {
    System.out.println("다운로드 중 에러 발생");
}
if(xml != null){
    parsing(xml);
}
else{
    System.out.println("데이터가 없습니다.");
}
}
```

기상청 날씨

```
private void parsing(String xml){
    try {
        if (xml != null) {
            //DOM - XML 파싱을 위한 객체를 생성
            DocumentBuilderFactory factory =
                DocumentBuilderFactory.newInstance();
            DocumentBuilder documentBuilder =
                factory.newDocumentBuilder();

            //문자열을 스트림으로 변환
            InputStream is = new ByteArrayInputStream(xml.getBytes());
            //파싱을 할 수 있도록 메모리에 모두 펼침
            Document doc = documentBuilder.parse(is);
            //루트를 찾아서 element에 저장
            Element element = doc.getDocumentElement();
            //tmx 태그를 전부 찾아서 items에 저장
            NodeList items = element.getElementsByTagName("tmx");
            int n = items.getLength();
        }
    }
}
```

기상청 날씨

추가

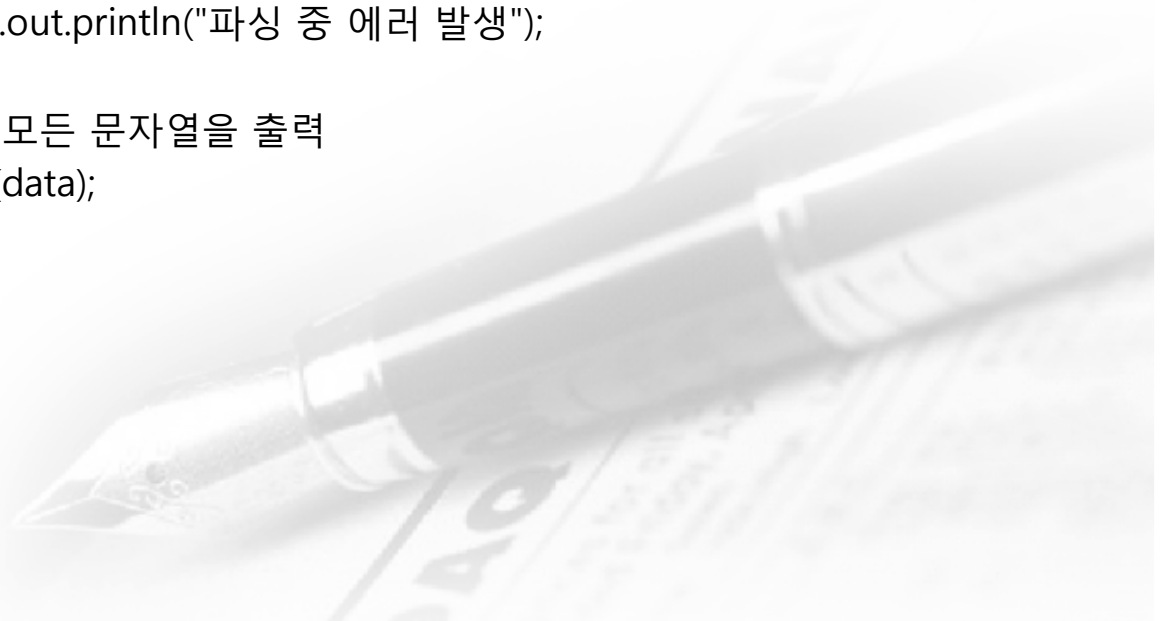
//items를 순회하면서 태그 안의 첫번째 값을 찾아서 data에

```
for (int i = 0; i < n; i++) {  
    Node item = items.item(i);  
    Node text = item.getFirstChild();  
    String itemValue = text.getNodeValue();  
    data.add(itemValue);  
}
```

```
    }  
} catch (Exception e) {  
    System.out.println("파싱 중 에러 발생");  
}  
//리스트에 저장된 모든 문자열을 출력  
System.out.println(data);
```

```
}
```

```
}
```



기상청 날씨

❖ 데이터를 받아와서 파싱하는 클래스(DomThreadMain.java)

```
package url;
```

```
public class DomThreadMain {
```

```
    public static void main(String[] args) {  
        DOMThread th = new DOMThread();  
        th.start();  
    }
```

```
}
```



XML 파싱

❖ SAX(Simple API for XML) 방식

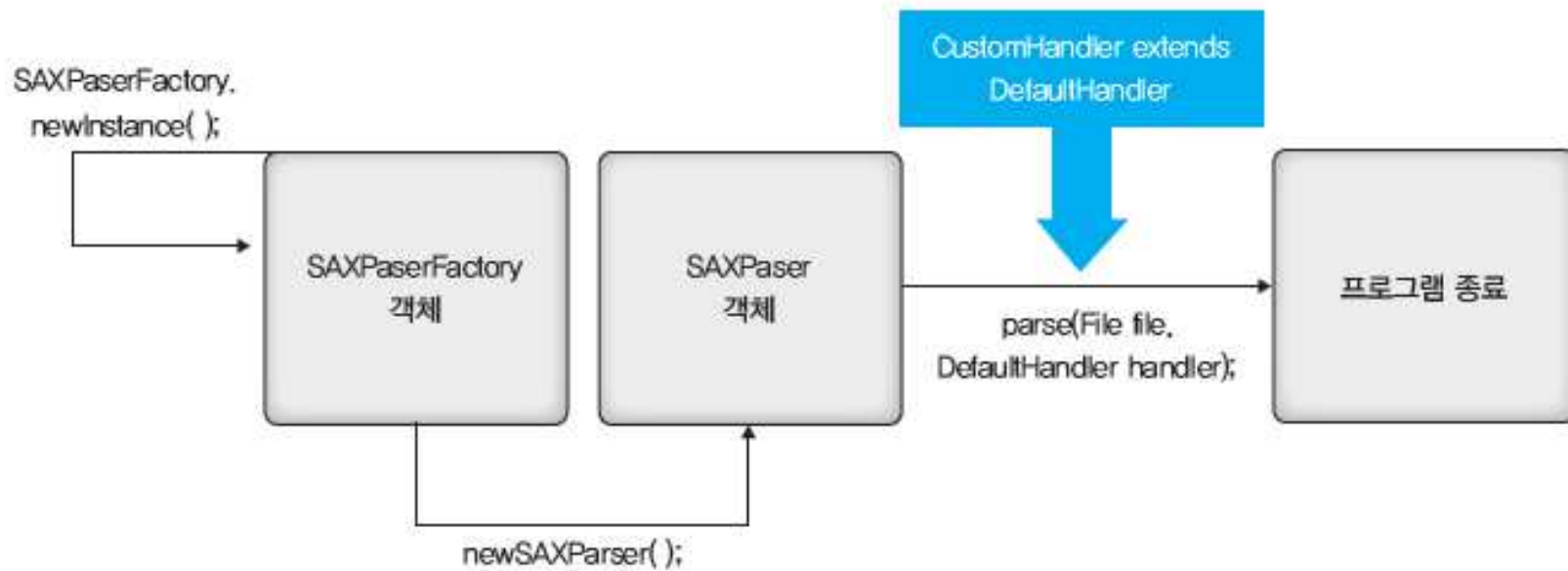
▷ SAX 파서의 특징

- ① SAX 파서는 이벤트 기반의 파싱을 지원
- ② 수행 시간이 빠름
- ③ 메모리 효율성이 높음
- ④ 구조화된 객체가 따로 필요
- ⑤ 순차적 파싱을 지원



XML 파싱

▶ SAX 파서 관련 주요 클래스와 메소드



XML 파싱

▶ SAX 파서 관련 주요 클래스와 메소드

① javax.xml.parsers.SAXParserFactory 클래스

▶ XML 문서를 파싱하는 SAXParser 객체를 만들기 위한 Factory 클래스

```
사용법 : public static SAXParserFactory newInstance( )  
        public static SAXParserFactory newInstance(String factoryClassName,ClassLoader classLoader)  
        SAXParserFactory [객체 이름] = SAXParserFactory.newInstance( );  
사용예 : SAXParserFactory factory = SAXParserFactory.newInstance();
```

▶ newSAXParser() : 현재 SAXParserFactory 객체에 설정된 값을 바탕으로 SAXParser 객체 생성

```
public abstract SAXParser newSAXParser() throws ParserConfigurationException,  
SXException
```

XML 파싱

▶ SAX 파서 관련 주요 클래스와 메소드

② javax.xml.parsers.SAXParser 클래스

- ▶ XML 문서를 파싱하기 위한 클래스
- ▶ SAXParser 객체를 만드는 방법

```
사용법 : SAXParser [객체 이름] = [SAXParserFactory 객체].newSAXParser( );  
사용예 : SAXParserFactory factory = SAXParserFactory.newInstance();  
         SAXParser parser = factory.newSAXParser();
```

- ▶ SAXParser 객체는 SAXParserFactory 클래스에서 제공한 newSAXParser() 메소드를 사용해서 반환

XML 파싱

▶ SAX 파서 관련 주요 클래스와 메소드

메소드	설명
getXMLReader	<ul style="list-style-type: none">• 선언부 : <code>public abstract XMLReader getXMLReader() throws SAXException</code>• 설명 : XMLReader 객체를 반환받기 위한 메소드다.
reset	<ul style="list-style-type: none">• 선언부 : <code>public void reset()</code>• 설명 : SAXParser 객체의 설정을 원래의 상태로 복구시킨다.
parse	<ul style="list-style-type: none">• 선언부 : <code>public void parse(File f, DefaultHandler dh) throws SAXException, IOException</code>• 설명 : <code>parse()</code> 메소드는 여러 형태로 오버로딩되어 있으며 <code>SAXException</code>과 <code>IOException</code> 예외를 던질 수 있다. 아래의 <code>parse()</code> 메소드 선언부에서는 예외 선언을 생략했으니 참고하도록 하자. 매개변수 <code>File f</code>와 <code>DefaultHandler</code> 객체를 사용하여 파싱한다. 매개변수 <code>f</code>와 XML 문자를 읽을 때마다 이벤트가 발생하며 이벤트에 따라서 <code>DefaultHandler</code> 객체의 메소드가 <code>SAXParser</code> 객체 내부에서 호출된다. 이벤트마다 발생하는 <code>DefaultHandler</code>의 메소드는 따로 설명한다.• 선언부 : <code>public void parse(InputStream is, DefaultHandler dh)</code> <code>public void parse(InputStream is, DefaultHandler dh, String systemId)</code>• 설명 : XML의 데이터를 읽을 때 <code>Input 스트림</code> 계열인 경우 사용하면 된다.• 선언부 : <code>public void parse(String uri, DefaultHandler dh)</code>• 설명 : XML의 데이터가 다른 시스템이나 네트워크상에 존재하고 있어 데이터의 위치가 <code>URI</code>로 표기되는 경우 이를 사용한다.

XML 파싱

▶ SAX 파서 관련 주요 클래스와 메소드

③ org.xml.sax.helpers.DefaultHandler 클래스

▶ DefaultHandler 클래스는 필요한 데이터를 받아온 후 ValueObject에 그 데이터를 설정하는 가장 중요한 클래스

▶ DefaultHandler 클래스를 사용하는 방법

```
사용법: DefaultHandler [객체 이름] = new DefaultHandler( );  
사용예: DefaultHandler handler = new DefaultHandler();  
        SAXParserFactory factory = SAXParserFactory.newInstance();  
        SAXParser parser = factory.newSAXParser();  
        parser.parse(new File("D:/source.xml", handler);
```

▶ DefaultHandler 클래스는 기본 생성자를 사용해서 인스턴스

▶ DefaultHandler 클래스에서 제공하는 메소드들은 모두 SAXException을 예외로 던짐

XML 파싱

▶ SAX 파서 관련 주요 클래스와 메소드

메소드	설명
characters	<ul style="list-style-type: none">• 선언부 : <code>public void characters(char[] ch, int start, int length)</code>• 설명 : XML 요소 내부에 데이터를 파싱할 때 이 <code>characters</code> 메소드가 내부적으로 호출된다. 즉, <code><name>Benjamin</name></code>이라는 내용을 <code>SAXParser</code> 객체가 파싱한다면 <code>name</code> 태그 요소 내부의 <code>Benjamin</code>이라는 값을 파싱할 때 <code>characters()</code> 메소드를 호출한다.
startDocument	<ul style="list-style-type: none">• 선언부 : <code>public void startDocument()</code>• 설명 : XML 문서를 맨 처음 파싱할 때 <code>startDocument()</code> 메소드가 내부적으로 호출된다.
endDocument	<ul style="list-style-type: none">• 선언부 : <code>public void endDocument()</code>• 설명 : <code>startDocument()</code> 메소드와 반대로 XML 문서 파싱이 끝나면 이 <code>endDocument()</code> 메소드가 내부적으로 호출된다.

XML 파싱

▶ SAX 파서 관련 주요 클래스와 메소드

startElement	<ul style="list-style-type: none">• 선언부 : <code>public void startElement(String uri, String localName, String qName, Attributes attributes)</code>• 설명 : XML의 시작 요소를 파싱할 때마다 <code>startElement</code> 메소드가 호출된다. <code><name>Benjamin</name></code> XML 데이터를 파싱한다면 <code><name></code> 태그를 파싱할 때 <code>startElement()</code> 메소드가 호출된다. 이 때 <code>startElement()</code> 메소드를 호출하면서 요소의 모든 정보값을 매개변수로 넘겨준다. 매개변수 <code>uri</code>는 XML의 Namespace URI를 의미한다. Namespace는 <code>xmlns</code>라는 속성 이름을 사용하며 <code><name xmlns='www.gilbut.co.kr'></code>이라는 태그를 파싱하면 <code>uri</code> 매개변수로는 <code>'www.gilbut.co.kr'</code>이라는 값이 넘어온다. <code>localname</code> 매개변수는 접두사를 제외한 노드의 이름이다. <code><name:first></code> 요소의 경우 <code>localName</code>은 <code>first</code>이다. 접두사를 포함한 노드의 이름은 <code>qName</code> 매개변수로 넘어온다. 또한 노드의 속성값들은 <code>Attributes</code> 객체로 넘어온다.
endElement	<ul style="list-style-type: none">• 선언부 : <code>public void endElement(String uri, String localName, String qName)</code>• 설명 : <code>startElement</code> 메소드와 반대로 XML의 종료 요소를 파싱할 때 <code>endElement()</code> 메소드가 호출된다. <code><name>Benjamin</name></code> XML 데이터를 파싱한다면 <code></name></code> 태그를 파싱할 때 <code>endElement()</code> 메소드가 호출된다.
error	<ul style="list-style-type: none">• 선언부 : <code>public void error(SAXParseException e)</code>• 설명 : <code>SAXParser</code> 객체로부터 회복 가능한 에러가 발생하면 <code>error()</code> 메소드가 호출된다.

HTML Parsing

HTML Parsing

- ❖ 안드로이드나 모바일 프로그래밍을 하다보면 API를 제공하지 않는 페이지를 처리해야 하는 경우가 많습니다.
- ❖ HTML Parsing을 수행해주는 라이브러리는 여러가지 존재합니다.
- ❖ jsoup 라이브러리는 파이썬의 BeautifulSoup와 유사한 형태의 라이브러리입니다.
- ❖ jsoup은 HTML 문서를 읽어들이고 후에 그 문서를 DOM 객체로 변환을 하게 됩니다.
- ❖ jsoup의 selector api를 이용해서 특정 Element에 접근을 할 수 있고, 해당 Element의 정보를 읽거나 수정할 수 있습니다.
- ❖ jsoup은 jquery의 selector와 비슷한 selector api를 제공하기 때문에 쉽게 사용할 수 있습니다.

jsoup

- ❖ jsoup은 기본적으로 HTML형식의 string을 넘겨주면 자바에서 사용할 수 있는 DOM 객체로 만들어 주는 parser의 역할
- ❖ 웹 페이지를 읽어들이는 기능까지 하는 라이브러리는 아님
- ❖ Jsoup.parse(String url, int timeoutMillisecons) api를 이용하면 URL로 부터 웹 페이지를 읽어와서 DOM 객체로 변환해 주긴 하지만 jsoup은 네트워크 라이브러리는 아니여서 해당 api를 사용하는 것은 비추천
- ❖ URLConnection 같은 네트워크 전용 API로 HTML을 읽어온 다음에 읽어온 string을 jsoup으로 변환시키는 방법을 사용하는 것이 바람직
- ❖ 읽어온 HTML을 jsoup document 객체로 변환시키는 방법

String html =(html 문서)

Document doc = Jsoup.parse(html);

jsoup

❖ 전통적인 방식의 Navigating api

- ✓ getElementById(), getElementsByTagName(), getElementsByClass(), getElementsByAttribute()와 같은 메소드를 이용하여 특정 Element를 찾아낼 수 있습니다.
- ✓ 조심해야 할 것은 getElementById 만 리턴 타입이 Element 이고, 나머지는 Elements
- ✓ id는 하나의 Element만 가지고 있는 고유 속성이기 때문에 단일 Element를 리턴
- ✓ Elements는 ArrayList<Element> 같은 것이 아니고 Elements라고 하는 객체가 따로 존재
- ✓ Elements는 List 인터페이스를 구현했기 때문에 일반 리스트 처럼 사용할 수 있습니다.
- ✓ Elements는 List이긴 하지만 jquery 객체 처럼 Element에 있는 대부분의 메소드를 가지고 있고 해당 메소드를 호출하면 Elements 내부에 있는 모든 Element에서 해당 메소드를 호출.
- ✓ Elements에는 addClass(String className) 메소드가 있는데 해당 메소드를 호출하게 되면 Elements가 포함하고 있는 모든 Element에 해당 클래스 이름을 추가.

jsoup

❖ jquery 방식의 Navigating api

- ✓ jquery처럼 \$('class#id') 와 비슷하게 사용할 수 있는 메소드가 jsoup에 있습니다.
- ✓ select 메소드를 사용하면 됩니다.
- ✓ Document doc = Jsoup.parse(html);
- ✓ Elements elements = doc.select(".class #id");
- ✓ select 메소드는 string 인자를 하나를 받고, 해당 인자는 jquery에서 사용되는 selector 와 비슷한 css query입니다.
- ✓ 리턴 값은 Elements
- ✓ Selector에 id 값을 넣어도 애초에 select 메소드의 리턴 타입은 Elements 이므로 Elements를 반환
- ✓ Element를 1개 가져오고 싶다면 elements.first()
- ✓ select 메소드는 getElementById와 getElementByTag와 getElementByClass 메소드를 여러번 호출해야 하는 일을 한번에 할 수 있게 해줍니다.
- ✓ select 메소드는 생각보다 굉장히 강력합니다.
- ✓ jsoup의 select syntax 문서를 보면 select("img[src\$=.png]")는 이미지 태그중 소스 파일 이름이 .png을 포함하고 있는 태그들만 추출
- ✓ \$=는 해당 문자열로 끝나는 attribute이 있는 확인할 수 있는 selector입니다.
- ✓ 비슷한 selector로 ^=는 해당 문자열로 시작하는 attribute가 있는지 찾고, *= 는 해당 문자열을 포함하는지 확인합니다

jsoup

✓ <http://www.bobaedream.co.kr/cyber/CyberCar.php?gubun=K&page=1>

The screenshot shows the Bobaedream website, which is a platform for buying and selling cars. The page is titled "보배드림 국내 1위 자동차소매점" (Bobaedream, Korea's #1 car dealership). The main navigation bar includes links for "사이버매장" (Cyber Store), "국산차" (Domestic Cars), "수입차" (Import Cars), "차량등록" (Vehicle Registration), "오토바이" (Motorcycles), "중고장터" (Used Car Market), "협력업체" (Cooperating Companies), "게시판" (Notice Board), and "자료실" (Data Room). Below the navigation bar, there are several filters and search options, including "차량" (Vehicle), "제조사" (Manufacturer), "모델" (Model), "세부모델" (Sub-model), "등급" (Grade), "세부등급" (Sub-grade), "연식" (Year), "선택" (Select), "지역" (Area), "시/도" (City/Province), and "구/군" (District/County). There are also buttons for "차량명" (Vehicle Name), "판매자" (Seller), "휴대폰" (Mobile Phone), "차량번호" (Vehicle Number), and "검색" (Search). A sidebar on the left lists various car categories: "사이버매장", "국산차매장", "수입차매장", "승용차매장", "스포츠카매장", "RV/SUV매장", "밴/승합차매장", "캠핑카매장", "튜닝카매장", "오토갤러리매장", and "차량등록/관리". The main content area displays a table of car listings. The table has columns for "전체" (All), "승용차" (Sedan), "스포츠카" (Sports Car), "RV/SUV", "밴/승합" (Van/MPV), "픽업/트럭" (Pickup/Truck), "버스" (Bus), "컨버터블" (Convertible), "캠핑카" (Camping Car), "오토카" (Auto), "튜닝카" (Tuning Car), and "회귀차" (Return Car). The table lists three cars: 1. 현대 그랜저IG 2.4 GDI 프리미엄 스페셜 (2017/04, 3,380만원, 이미지 인천 서구, 등록 07.06, 조회 6,423). 2. 스페이스 이등 업무차 (2017/07, 5,950만원, 한인성, 충남 천안시, 등록 07.06, 조회 7,888). 3. 르노삼성 뉴SM5 플레티움 TCE 1.6 터보 (2014/07, 1,699만원, 전재화, 대구 동구, 등록 07.06, 조회 2,200). A sidebar on the right contains a "현대캐피탈 다이렉트 중고차론" (Hyundai Capital Direct Used Car Loan) advertisement with a phone number 02-784-2329 and a link to "인터넷으로 신청하기" (Apply online). Below the advertisement is a "점한차량 (0)" (Pointed Vehicle (0)) section with links for "사고이력조회" (Accident History Check), "보험료계산" (Insurance Premium Calculation), "할부한도조회" (Lease Limit Check), and "내차팔기" (Sell My Car). At the bottom of the sidebar are social media icons for Facebook, YouTube, and Instagram.

전체	승용차	스포츠카	RV/SUV	밴/승합	픽업/트럭	버스	컨버터블	캠핑카	오토카	튜닝카	회귀차
	현대 그랜저IG 2.4 GDI 프리미엄 스페셜 ▶▶1인 소유/무 사고/최저가/추가옵션/신차상태 그대로 자동 가솔린 7,700 km	2017/04 (2018년형)	3,380만원	이미지 인천 서구 등록 07.06 조회 6,423							
	스페이스 이등 업무차 2017 신차 출고, 떠나기만 하면 되는 이등 캠핑 자동 디젤 400 km	2017/07	5,950만원	한인성 충남 천안시 등록 07.06 조회 7,888							
	르노삼성 뉴SM5 플레티움 TCE 1.6 터보 워크LS-207/KEBB 일체형/OTEM바다/피코가변 자동 가솔린 76,500 km	2014/07	1,699만원	전재화 대구 동구 등록 07.06 조회 2,200							

jsoup

- ✓ 보배드림 페이지에서 자동차 정보 가져오기

```
{transmission=자동, distance=79,000 km, engine=디젤, price=1,750, title=기아 봉고3 워크스루밴}  
{transmission=자동, distance=159,066 km, engine=LPG 일반인구입, price=399, title=르노삼성 뉴 SM5 LPLi}  
{transmission=자동, distance=49,000 km, engine=가솔린, price=568, title=기아 올 뉴 모닝 1.0 가솔린 디럭스}  
{transmission=자동, distance=168,800 km, engine=LPG 일반인구입, price=739, title=현대 YF쏘나타 2.0 Y2}  
{transmission=자동, distance=90,700 km, engine=디젤, price=1,150, title=현대 뉴 i30 1.6 VGT 익스트림}  
{transmission=수동, distance=42,610 km, engine=디젤, price=4,980, title=현대 뉴 카운티 장축 어린이 39인:  
{transmission=자동, distance=16,000 km, engine=가솔린, price=8,800, title=제네시스 EQ900 5.0 GDi AWC}  
{transmission=수동, distance=145,000 km, engine=디젤, price=4,000, title=현대 카운티 캠핑카}  
{transmission=자동, distance=38,000 km, engine=가솔린, price=2,750, title=현대 뉴 제네시스 G330 모던}  
{transmission=자동, distance=76,090 km, engine=가솔린, price=1,700, title=GM대우 알페온 EL240 디럭스}  
{transmission=수동, distance=81,765 km, engine=가솔린, price=1,120, title=현대 제네시스 쿠페 200 터보 F}  
{transmission=수동, distance=94,000 km, engine=가솔린, price=899, title=현대 제네시스 쿠페 200 터보 P}  
{transmission=자동, distance=2,200 km, engine=가솔린, price=9,190, title=제네시스 EQ900 3.3 T-GDi AWC}  
{transmission=수동, distance=9,193 km, engine=디젤, price=금용리스, title=기아 봉고3 냉동 탑차}  
{transmission=자동, distance=263,629 km, engine=가솔린, price=330, title=현대 에쿠스 JS 350 밸류}  
{transmission=자동, distance=194,983 km, engine=가솔린, price=1,030, title=현대 제네시스 BH330 럭셔리}  
{transmission=자동, distance=77,473 km, engine=가솔린, price=1,920, title=기아 더 뉴 K7 2.4 GDI 프레스
```

jsoup

❖ 프로젝트의 build path에 jsoup 라이브러리를 추가
(www.mvnrepository.com 에서 검색한 후 다운로드)

❖ 소스 코드 작성 – HTMLMain

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
```

```
import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import org.jsoup.nodes.Element;
import org.jsoup.select.Elements;
```


jsoup

```
public class HTMLMain {  
    public static void main(String[] args) {  
        StringBuilder sBuffer = new StringBuilder();  
        String html = "";  
        try {  
            //데이터를 다운로드 받을 주소 생성  
            String urlAddr =  
"http://www.bobaedream.co.kr/cyber/CyberCar.php?gubun=K&page=1";  
  
            URL url = new URL(urlAddr);  
            //주소와 연결  
            HttpURLConnection conn = (HttpURLConnection)  
url.openConnection();
```

jsoup

```
if (conn != null) {  
    conn.setConnectTimeout(20000);  
    conn.setUseCaches(false);  
    if (conn.getResponseCode() == HttpURLConnection.HTTP_OK) {  
        InputStreamReader isr = new  
InputStreamReader(conn.getInputStream());  
        BufferedReader br = new BufferedReader(isr);
```



jsoup

// 줄 단위로 읽어서 sBuffer에 저장

```
while (true) {  
    String line = br.readLine();  
    if (line == null) {  
        break;  
    }  
    sBuffer.append(line);  
}  
br.close();  
conn.disconnect();  
}
```

jsoup

```
}
```

```
        // 전부 읽었으면 String으로 변환  
        html = sBuffer.toString();  
    } catch (Exception e) {  
        System.out.println("다운로드 중 에러 발생");  
    }  
}
```



jsoup

//html 문자열을 메모리에 펼침

```
Document doc = Jsoup.parse(html);
```

//펼쳐진 데이터에서 클래스 이름이 carinfo 인 데이터 찾아오기

```
Elements root = doc.getElementsByClass("mode-cell title");
```

//클래스가 price 인 데이터 찾아오기

```
Elements prices = doc.getElementsByClass("price");
```

```
//System.out.println(prices);
```

//파싱한 결과를 저장할 리스트 생성

```
List<Map<String, String>> list = new ArrayList<Map<String,  
String>>();
```

jsoup

```
for (int i = 1; i < root.size(); i++) {  
    Element element = root.get(i);  
    //carinfo 클래스 안에 있는 요소 중에서 클래스가  
title 인 데이터 전부 찾아오기  
    Elements titles = element.getElementsByTag("a");  
    //price를 순회하면서 em 태그의 데이터 찾아오기  
    Elements em = prices.get(i).getElementsByTag("em");  
    //하나의 데이터를 저장할 맵을 생성  
    Map<String, String>map = new  
HashMap<String,String>();  
    //title 클래스의 첫번째 데이터의 문자열을 title이라  
는 키에 저장  
    map.put("title", titles.get(0).text());  
    list.add(map);  
}
```

jsoup

```
        for (Map<String, String> map : list) {  
            System.out.println(map);  
        }  
    }  
}
```



EMAIL 전송

1. CommonsEmail

- ▶ CommonsEmail 컴포넌트 : 이메일에 관련된 컴포넌트
자바를 사용해서 이메일을 보낼 수 있는 각종 API를 제공
- ▶ CommonsEmail 컴포넌트는 서버 프로그래머들이 많이 사용
- ▶ 다음과 같은 몇몇 목적으로 사용
 - ✓ •서비스 사용자들에게 이벤트 혹은 정보를 발송하기 위한 광고 메일
 - ✓ •암호를 잊어버린 사용자에게 암호를 전달해주기 위한 메일
 - ✓ •서버에 문제가 발생했을 때 관리자에게 알려주기 위한 알람성 메일
- ▶ CommonsEmail 라이브러리는 이메일을 전송하는 기능만 제공할 뿐 받는 기능은 제공하지 않음

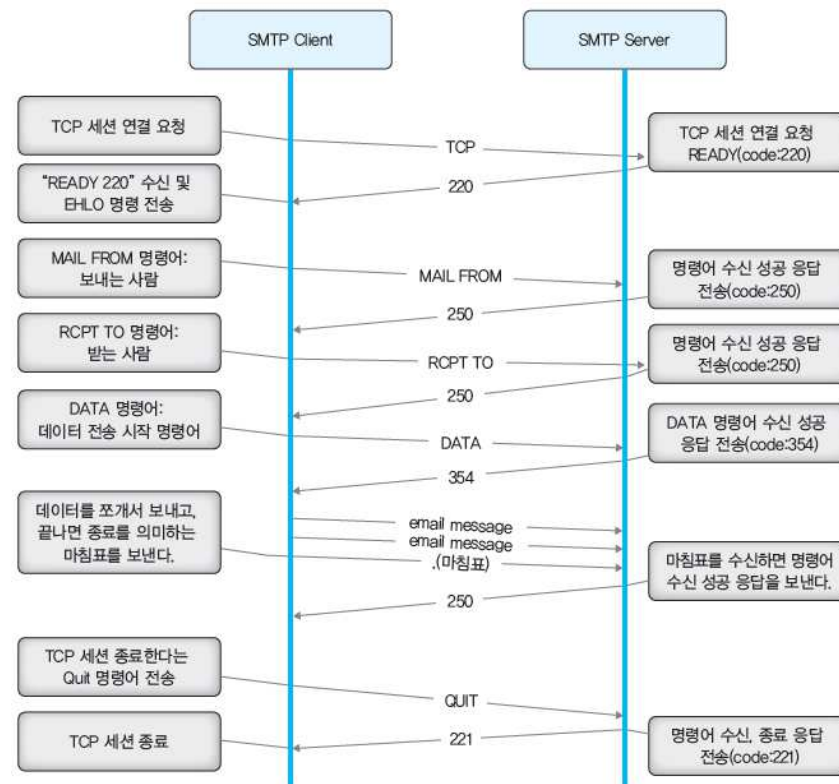
1. CommonsEmail

▶ SMTP 프로토콜

- ▶ SMTP 프로토콜(Simple Mail Transfer Protocol)
- ▶ 인터넷상에서 이메일을 전송하기 위해서 사용되는 통신 규약 중에 하나
- ▶ CommonsEmail 라이브러리를 이용해서는 이메일을 보내는 기능만 구현
- ▶ 클라이언트는 SMTP 프로토콜에 의해서 미리 정의된 절차에 따라서 명령어를 전송
서버는 해당 명령어가 성공적으로 처리되었을 때
사전에 정의된 응답 코드에 따라서 다음 절차를 수행

1. CommonsEmail

▶ SMTP 프로토콜



1. CommonsEmail

▶ SMTP 프로토콜

- ▶ SMTP 프로토콜을 지원하는 Email 서버가 필요
- ▶ 리눅스나 유닉스 환경의 서버를 운영하고 있다면 SendMail과 같은 데몬을 설치하여 Mail 서버 구축
- ▶ 윈도우 환경의 서버를 운영하고 있다면 윈도우에서 제공하는 메일 서버를 설치해서 구성
- ▶ Gmail 계정이 있으면 접근 가능 한 SMTP 서버의 주소

SMTP Server address : smtp.gmail.com

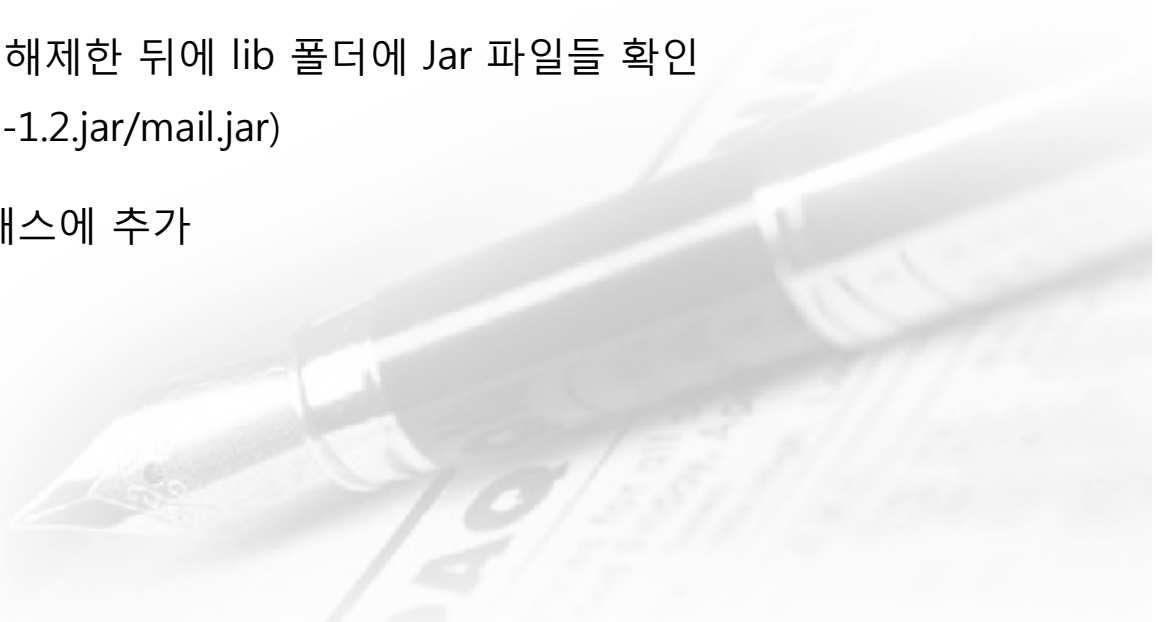
SMTP Server port : 587

SMTP Server account : 여러분의 google 가입 계정(아이디와 비밀번호)

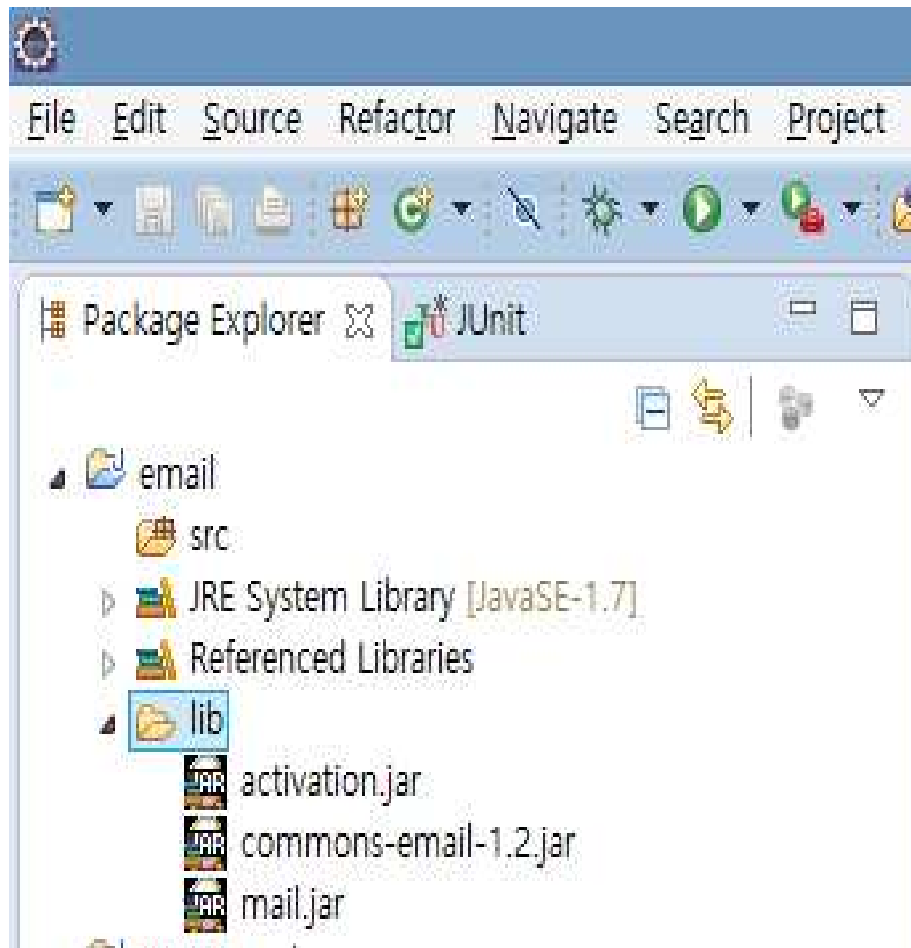
1. CommonsEmail

▶ 환경 설정

- ① CommonEmail 컴포넌트를 이용하여 이메일을 보내기 위해서는 3개의 라이브러리 파일 필요
'activation.jar', 'common_email_1.2.jar' 그리고 'mail.jar' 파일
 - CommonEmail : http://commons.apache.org/email/download_email.cgi
 - Javamail : <http://www.oracle.com/technetwork/java/index-138643.html>
 - JAF : <http://www.oracle.com/technetwork/java/jaf11-139815.html>
- ② 다운로드한 zip 파일의 압축을 해제한 뒤에 lib 폴더에 Jar 파일들 확인
(activation.jar/commons-email-1.2.jar/mail.jar)
- ③ 압축을 푼 jar 파일들을 빌드 패스에 추가

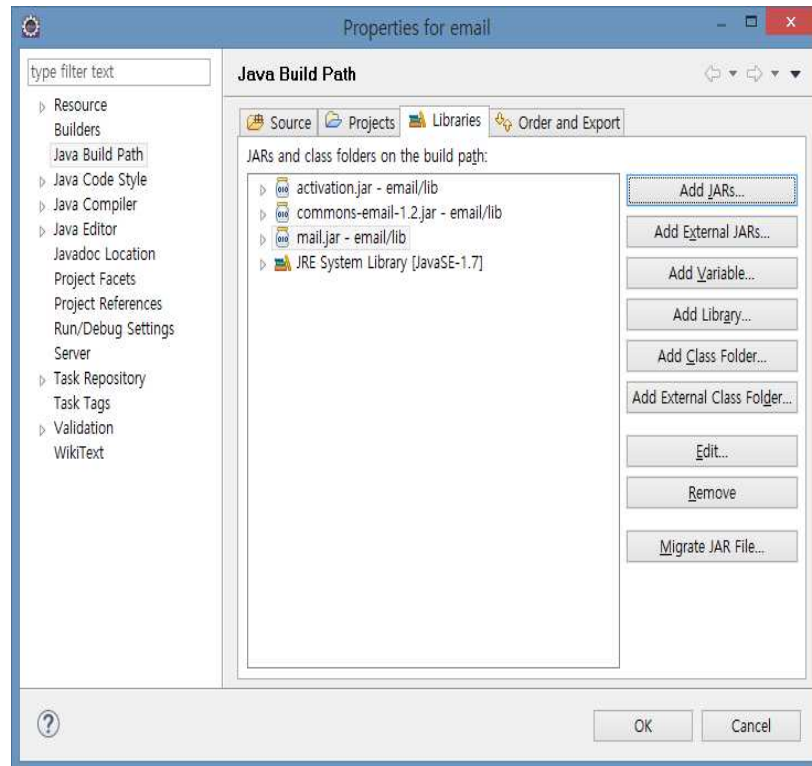


1. CommonsEmail



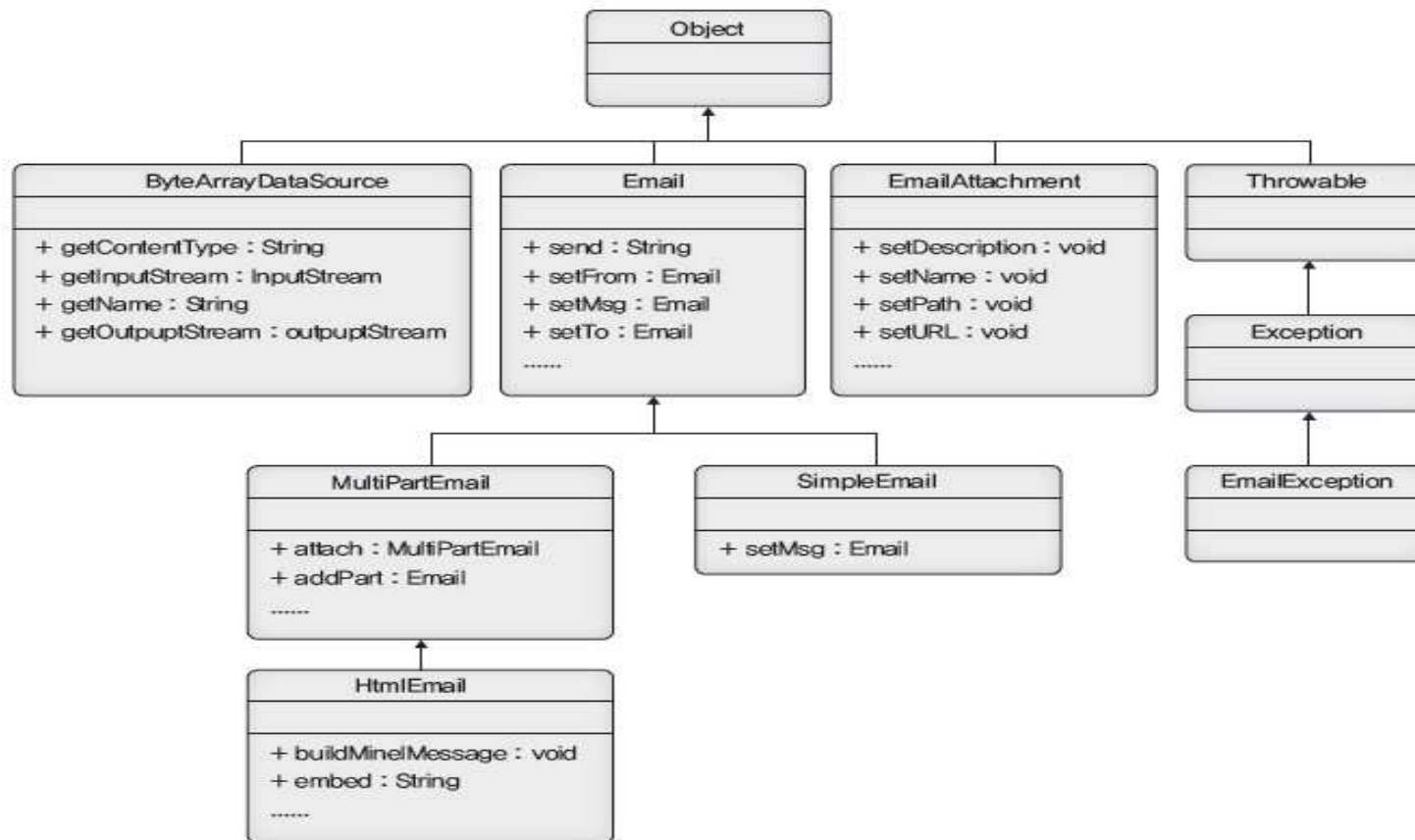
- ④ 폴더를 새로 생성해서
위의 3개의 jar 파일을 복사

1. CommonsEmail



- 5 빌드 패스에 3개의 라이브러리를 추가
마우스 오른쪽 버튼
Properties → Java Build Path →
Libraries 탭 → <Add JARs> 버튼
해당 디렉토리에서 jar 파일을 선택한 다음,
<JAR 파일 선택 및 완료> 버튼을 클릭

2. CommonsEmail API



2. CommonsEmail API

▶ Email 클래스

- ▶ **Email 클래스** : 이메일을 보내기 위한 여러 정보들
즉, 받는 사람, 보내는 사람, 제목 그리고 메시지 등을
설정할 수 있는 API와 Email 전송 API를 제공
- ▶ 다른 Email 클래스들의 상위 클래스이므로 Email 클래스에서 제공되는 메소드들은
다른 하위 클래스들에게 상속
- ▶ Email 클래스는 abstract 키워드로 선언된 클래스이므로 Email 클래스를 사용하여
직접 인스턴스 생성 불가
- ▶ Email 추상 클래스를 구현한 SimpleEmail이나 MultiPartEmail 클래스의 객체를 생
성해서 사용

2. CommonsEmail API

▶ Email 클래스

메소드	설명
addBcc	<ul style="list-style-type: none">• 선언부 : <code>public Email addBcc(String email)</code> <code>public Email addBcc(String email, String name)</code> <code>public Email addBcc(String email, String name, String charset)</code>• 설명 : BCC란 숨은 참조를 의미한다. 숨은 참조에 포함된 사람도 이메일을 전송받으며 그 사람들은 단지 이메일을 참고할 뿐이다. 위의 메소드들을 사용하면 숨은 참조를 추가할 수 있다. 매개변수 email은 이메일 주소를 의미하며 name은 그 사람의 이름을 의미한다. charset은 문자열 인코딩 셋을 의미한다.
addCc	<ul style="list-style-type: none">• 선언부 : <code>public Email addCc(String email)</code> <code>public Email addCc(String email, String name)</code> <code>public Email addCc(String email, String name, String charset)</code>• 설명 : Cc란 참조를 의미한다. Cc에 포함된 사람은 이메일을 단지 참고하면 된다. 참조될 사람의 이메일을 <code>addCc()</code> 메소드를 사용하여 추가한다.
addHeader	<ul style="list-style-type: none">• 선언부 : <code>public void addHeader(String name, String value)</code>• 설명 : SMTP 프로토콜의 헤더에 특정 값을 추가할 때 사용한다. 매개변수 name은 헤더 이름이며 value는 그 헤더의 값을 의미한다.

2. CommonsEmail API

▶ Email 클래스

addReplyTo	<ul style="list-style-type: none">• 선언부 : <code>public Email addReplyTo (String email)</code> <code>public Email addReplyTo (String email, String name)</code> <code>public Email addReplyTo (String email, String name, String charset)</code>• 설명 : 이메일을 읽은 뒤 응답을 위해 받는 사람들을 추가할 때 사용하는 메소드다.
addTo	<ul style="list-style-type: none">• 선언부 : <code>public Email addTo (String email)</code> <code>public Email addTo (String email, String name)</code> <code>public Email addTo (String email, String name, String charset)</code>• 설명 : 이메일을 받는 사람의 이메일 주소를 넣는다. 여러 사람에게 보낼 수 있으므로 계속 메소드를 호출해도 무방하다.
send	<ul style="list-style-type: none">• 선언부 : <code>public String send()</code>• 설명 : Email 객체에 여러 정보를 설정하고 그 데이터들을 전송할 때 호출하는 메소드다. 이 메소드가 호출되면 실제 이메일이 전송된다.

2. CommonsEmail API

▶ Email 클래스

setAuthentication	<ul style="list-style-type: none">• 선언부 : <code>public void setAuthentication (String username, String password)</code>• 설명 : SMTP 서버와 연결할 때 SMTP 서버에 접속하기 위한 인증을 받아야 한다. 이때 <code>setAuthentication()</code> 메소드를 사용하여 SMTP 계정 정보인 사용자 아이디와 암호를 설정하면 된다. <code>setAuthentication()</code> 메소드에 의해서 설정된 계정은 <code>send()</code> 메소드에 의해서 SMTP 서버 인증을 받을 때 사용된다.
setFrom	<ul style="list-style-type: none">• 선언부 : <code>public Email setFrom(String email)</code> <code>public Email setFrom(String email, String name)</code> <code>public Email setFrom(String email, String name, String charset)</code>• 설명 : 보내는 사람의 이메일 주소를 넣는다. 보내는 사람은 항상 한 명뿐이므로 <code>addTo()</code> 메소드와 다르게 <code>setFrom()</code> 메소드는 계속 호출하더라도 마지막에 호출된 값에 의해서 보낸 사람의 정보가 설정된다.
setHostName	<ul style="list-style-type: none">• 선언부 : <code>public void setHostName(String aHostName)</code>• 설명 : SMTP 서버의 Host 이름이나 IP 주소를 입력하도록 한다.

2. CommonsEmail API

▶ Email 클래스

setSmtpPort	<ul style="list-style-type: none">• 선언부 : <code>public void setSmtpPort(int aPortNumber)</code>• 설명 : SMTP 기본 포트는 465번이지만 SMTP 서버의 설정에 따라서 port 번호가 다를 수 있다. 이 메소드를 사용해서 포트를 변경하지 않으면 Email 클래스는 기본 포트 465번을 이용하여 SMTP 서버에 연결하려고 한다.
setSSL	<ul style="list-style-type: none">• 선언부 : <code>public void setSSL(boolean ssl)</code>• 설명 : 메일 클라이언트와 메일 서버 사이에 데이터를 암호화하기 위해서 사용한다.
setSubject	<ul style="list-style-type: none">• 선언부 : <code>public Email setSubject(String subject)</code>• 설명 : 보내고자 하는 메일의 제목을 입력한다.
setMsg	<ul style="list-style-type: none">• 선언부 : <code>public static Email set msg(String msg)</code>• 설명 : 메일 본문을 사용한다. 개행 문자(\n)를 사용해서 줄바꿈을 한다.
setDebug	<ul style="list-style-type: none">• 선언부 : <code>public void setDebug(boolean d)</code>• 설명 : 디버깅 관련 정보를 출력하는 메소드다. true인 경우 디버깅 정보가 출력된다.

2. CommonsEmail API

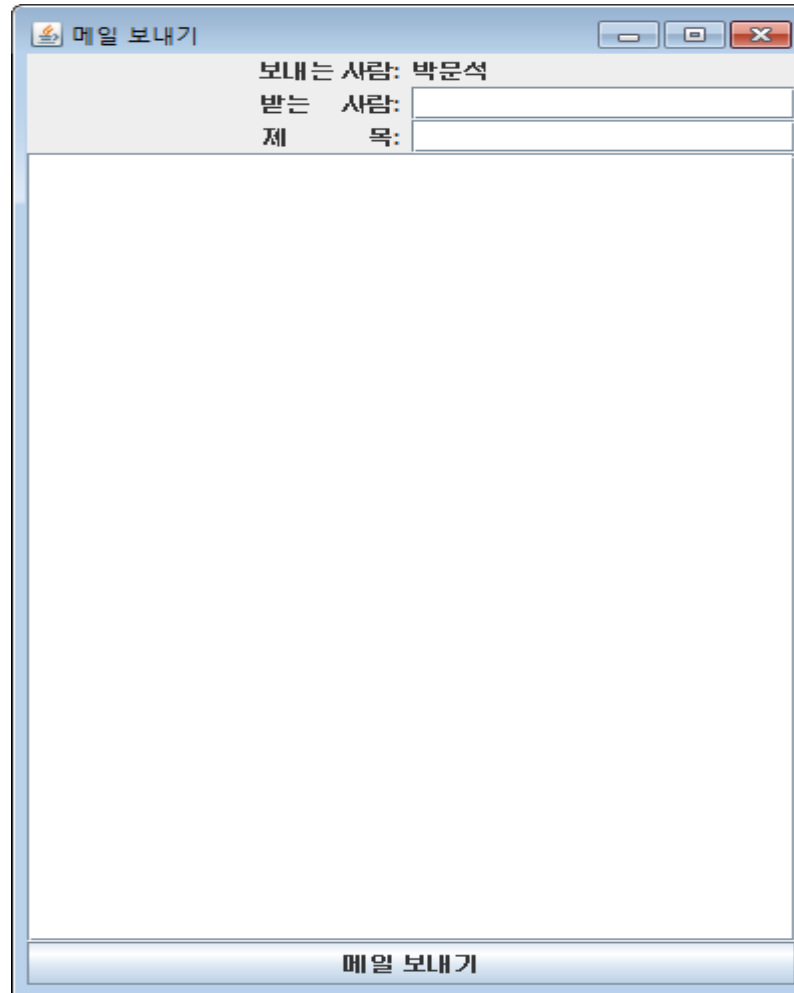
▶ 텍스트 전송을 위한 SimpleEmail 클래스

- ▶ SimpleEmail 클래스 : 특별한 추가 기능 없이 순수한 텍스트 기반의 이메일을 전송하고자 할 때 사용
- ▶ SimpleEmail 객체를 사용하는 방법

```
사용법 : SimpleEmail [객체 이름] = new SimpleEmail();  
사용예 : SimpleEmail email = new SimpleEmail();
```

- ▶ SimpleEmail 클래스는 public으로 선언된 생성자들이 존재하므로 new 키워드를 사용하여 객체를 인스턴스
- ▶ 매개변수가 없는 기본 생성자만 제공하고 있으므로 매우 간단

3. 텍스트 메일 보내기



메일 보내기

보내는 사람: 박문석

받는 사람:

제목:

메일 보내기

3. 텍스트 메일 보내기

1.GUI 만들기

```
public class MailView extends JFrame {  
    public MailView(){  
        setTitle("메일 보내기");  
        setBounds(100,100,400,600);  
        setDefaultCloseOperation(EXIT_ON_CLOSE);  
        JPanel northPanel = new JPanel();  
        northPanel.setLayout(new GridLayout(3,2));  
        JLabel lblSend = new JLabel("보내는 사람: ", JLabel.RIGHT);  
        JLabel sendEmail = new JLabel("박문석");  
        JLabel lblReceive = new JLabel("받는 사람: ",  
JLabel.RIGHT);  
        JTextField receiveEmail = new JTextField(30);  
        JLabel lblTitle = new JLabel("제목: ",  
JLabel.RIGHT);  
        JTextField txtTitle = new JTextField(30);  
        northPanel.add(lblSend);  
        northPanel.add(sendEmail);  
        northPanel.add(lblReceive);  
        northPanel.add(receiveEmail);
```


3. 텍스트 메일 보내기

```
northPanel.add(lblTitle);  
northPanel.add(txtTitle);  
add("North", northPanel);
```

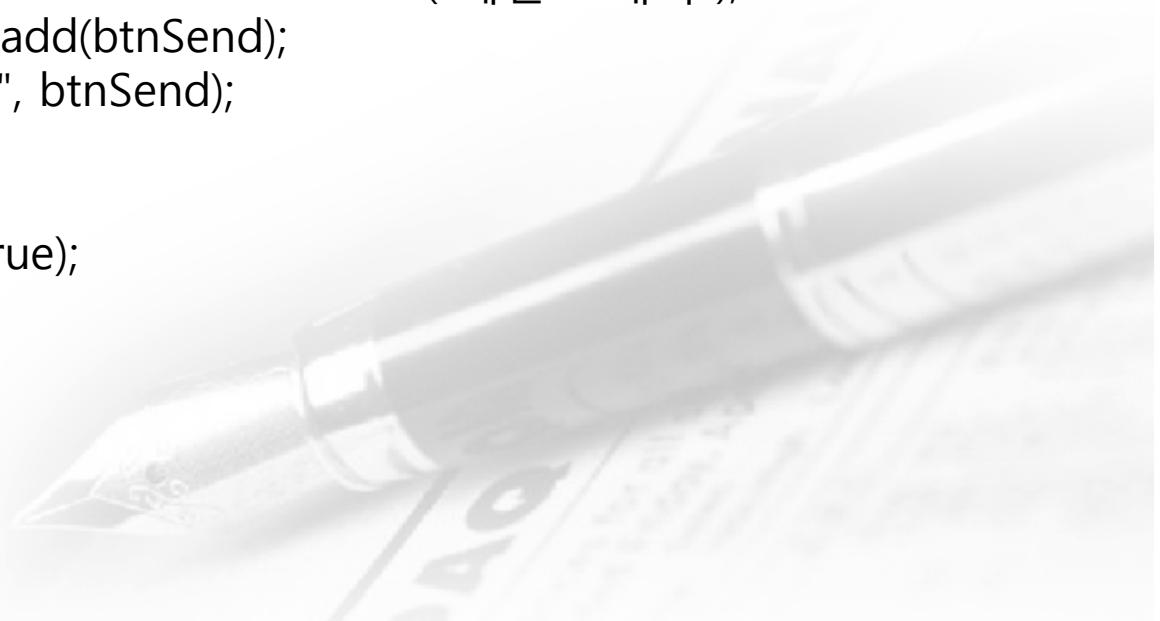
```
JTextArea content = new JTextArea(20,20);  
JScrollPane sp = new JScrollPane(content);  
add(sp);
```

```
JPanel southPanel = new JPanel();  
JButton btnSend = new JButton("메일 보내기");  
southPanel.add(btnSend);  
add("South", btnSend);
```

```
setVisible(true);
```

```
}
```

```
}
```



3. 텍스트 메일 보내기

2.Main 클래스

```
public class Main {  
    public static void main(String[] args) {  
        new MailView();  
    }  
}
```

3.3개의 jar 파일을 BuildPath에 추가

activation.jar
commons-email.jar
mail.jar



3. 텍스트 메일 보내기

4.네이버 메일인 경우 환경설정 메뉴에서 SMTP 설정

NAVER 메일

환경 설정 | 메일로 돌아가기

기본 환경 설정 메일함 관리 메일 자동 분류 서명/빠른답장 설정 부재 중 설정
새 메일 알림 설정 스팸 설정 외부 메일 가져오기 **✓ POP3/IMAP 설정** 단축키

POP3/SMTP 설정 **IMAP/SMTP 설정**

스마트폰에서 네이버 메일을 더욱 잘 관리하고 쓸 수 있도록 IMAP/SMTP를 설정합니다.

ggangpae3님은 현재 IMAP/SMTP를 사용하고 있지 않습니다.
아래에서 설정을 변경하시면 IMAP/SMTP를 이용하실 수 있습니다.

IMAP/SMTP 사용 ☒ 사용함 ☐ 사용 안 함?

기본 설정으로 **확인** 취소

· **스마트폰 메일 애플리케이션 계정 설정**
스마트폰의 메일 계정 설정에 아래와 같이 등록해 주세요.

IMAP 서버명 : imap.naver.com	SMTP 서버명 : smtp.naver.com	IMAP 포트 : 993, 보안연결(SSL) 필요
SMTP 포트 : 587, 보안 연결(TLS) 필요 (TLS가 없는 경우 SSL로 연결)	아이디 : ggangpae3	비밀번호 : 네이버 로그인 비밀번호

TIP 설정가이드 동영상 보기 IMAP이란 무엇인가요? 네이버 메일 IMAP에서 지원하는 프로그램은 무엇인가요?

용량 99KB / 5GB | 환경설정 | 스팸설정 | 위젯사용하기 | 모바일 메일

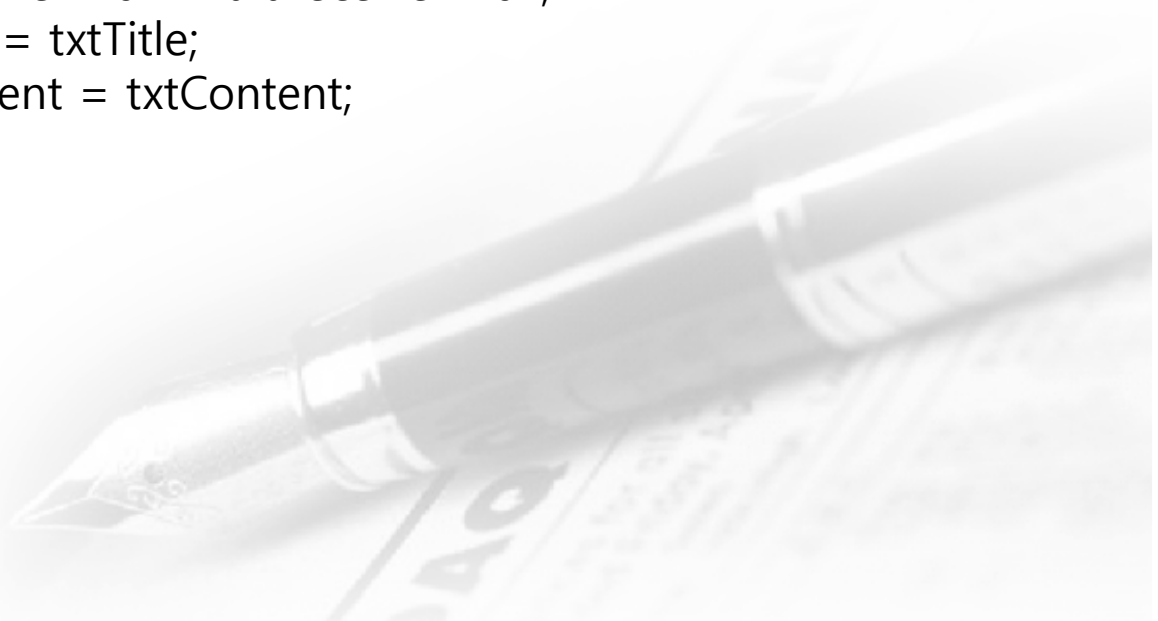
Copyright © NAVER Corp. All Rights Reserved.

메일Lite 기기 | 스팸정책 | 공지사항 | 메일 고객센터

3. 텍스트 메일 보내기

5.EventHandler 만들기

```
public class EventHandler implements ActionListener {  
    private JTextField txtReceiveEmail;  
    private JTextField txtTitle;  
    private JTextArea txtContent;  
  
    public EventHandler(JTextField txtReceiveEmail, JTextField  
txtTitle, JTextArea txtContent) {  
        super();  
        this.txtReceiveEmail = txtReceiveEmail;  
        this.txtTitle = txtTitle;  
        this.txtContent = txtContent;  
    }  
}
```



3. 텍스트 메일 보내기

```
@Override
public void actionPerformed(ActionEvent e) {
    String receiveEmail = txtReceiveEmail.getText().trim();
    if(receiveEmail.length() == 0){
        JOptionPane.showMessageDialog(null, "받는 분의
이메일 주소가 없습니다.");
        return;
    }
    String title = txtTitle.getText().trim();
    if(title.length() == 0){
        title = "무제";
    }
    String content = txtContent.getText().trim();
    if(content.length() == 0){
        content = "내용";
    }

    SimpleEmail simpleEmail = new SimpleEmail();
```

3. 텍스트 메일 보내기

```
// Smtplib 서버 연결 설정.
simpleEmail.setHostName("smtp.naver.com");
simpleEmail.setSmtplibPort(587);
simpleEmail.setAuthentication("ggangpae3", "wnddkd");
String rt = "";
try
{
    //Smtplib SSL, TLS 설정
    simpleEmail.setSSL(true);
    simpleEmail.setTLS(true);
    simpleEmail.setCharset("utf-8");

    //받는 사람 설정
    simpleEmail.addTo(receiveEmail, "받는 이", "utf-8");

    //보내는 사람 설정
    simpleEmail.setFrom("ggangpae3@naver.com", "박문석", "utf-8");

    //제목 설정
    simpleEmail.setSubject(title);
```

3. 텍스트 메일 보내기

```
        //본문 설정
        simpleEmail.setMsg(content);
        rt = simpleEmail.send();
        JOptionPane.showMessageDialog(null, "메일 보내
기 성공");

        txtReceiveEmail.setText("");
        txtTitle.setText("");
        txtContent.setText("");

    }
    catch (Exception exception)
    {
        JOptionPane.showMessageDialog(null,
exception.getMessage(), "메일 보내기 실패", JOptionPane.ERROR_MESSAGE);
    }

}

}
```

3. 텍스트 메일 보내기

6.MailView의 생성자에서 메일 보내기 버튼 이벤트 핸들러 설정

```
EventHandler handler = new EventHandler(receiveEmail, txtTitle, content);  
btnSend.addActionListener(handler);
```



4. 첨부 파일 이메일

▶ 메시지와 파일을 함께 전송하는 MultiPartEmail 클래스

- ▷ **MultiPartEmail 클래스** : 메일을 전송할 때 여러 개의 파일을 첨부해서 보낼 수 있는 기능을 제공하는 클래스
- ▷ **Multi-Part 전송** : 첨부 파일을 전송할 때 메시지와 파일을 여러 개로 분할해서 전송하는 방식
- ▷ MultiPartEmail 객체를 인스턴스하는 방법

```
사용법: MultiPartEmail [객체 이름] = new MultiPartEmail();  
사용예: MultiPartEmail email = new MultiPartEmail();
```

4. 첨부 파일 이메일

▶ 메시지와 파일을 함께 전송하는 MultiPartEmail 클래스

메소드	설명
attach	<ul style="list-style-type: none">• 선언부 : <code>public MultiPartEmail attach(EmailAttachment attachment)</code> <code>public MultiPartEmail attach(DataSource ds, String name, String description)</code> <code>public MultiPartEmail attach(DataSource ds, String name, String description, String disposition)</code> <code>public MultiPartEmail attach(URL url, String name, String description)</code>• 설명 : 파일을 첨부하기 위한 메소드들이다. <code>attach()</code> 메소드가 호출되면 첨부 파일은 <code>MultiPartEmail</code> 객체의 속성으로 설정된다. 그리고 <code>send()</code> 메소드 호출 시 전송된다. 또한 다양한 형태의 첨부 파일을 처리할 수 있도록 <code>attach()</code> 메소드들이 오버라이딩되어 있다. 보통은 <code>CommonsEmail</code> 컴포넌트에서 제공하는 <code>EmailAttachment</code> 객체를 이용해서 <code>attach()</code> 메소드를 호출한다. 매개변수 중 <code>name</code>은 첨부 파일의 이름을 의미하며 <code>description</code>은 설명, 그리고 <code>disposition</code>은 어떤 성격의 첨부 파일인지 의미한다.

4. 첨부 파일 이메일

▶ 첨부 파일을 위한 EmailAttachment 클래스

- ▷ **EmailAttachment 클래스** : 이메일에 첨부 파일을 추가해서 보낼 때 사용
첨부 파일을 추상화한 클래스로서 반드시 MultipartEmail 클래스와 같이 사용
- ▷ EmailAttachment 클래스를 인스턴스화 하는 방법

사용법: EmailAttachment [객체 이름] = new EmailAttachment();

사용예: EmailAttachment attachment= new EmailAttachment();



4. 첨부 파일 이메일

▶ 첨부 파일을 위한 EmailAttachment 클래스

메소드	설명
getDescription/ setDescription	<ul style="list-style-type: none">• 선언부 : public String getDescription() public void setDescription(String desc)• 설명 : 첨부 파일에 대한 설명을 EmailAttachment 객체에 설정(set)하거나 설정한 것을 받아오는(get) 메소드
getDisposition/ setDisposition	<ul style="list-style-type: none">• 선언부 : public String getDisposition() public void setDisposition(String aDisposition)• 설명 : 첨부 파일에 대한 성격을 설정(set)하거나 설정된 것을 받아오는(get) 메소드
getName/ setName	<ul style="list-style-type: none">• 선언부 : public String getName() public void setName(String aName)• 설명 : EmailAttachment 객체에 이름을 설정하거나 받아오는 메소드

4. 첨부 파일 이메일

▶ 첨부 파일을 위한 EmailAttachment 클래스

getPath/ setPath	<ul style="list-style-type: none">• 선언부 : <code>public String getPath()</code> <code>public void setPath(String aPath)</code>• 설명 : EmailAttachment 객체에 Path를 지정하면 Path에 위치한 파일이 첨부 파일이 된다. 실제 첨부 파일을 설정하는 메소드이므로 EmailAttachment 클래스에서 가장 중요한 메소드다.
getURL/ setURL	<ul style="list-style-type: none">• 선언부 : <code>public URL getURL()</code> <code>public void setURL(URL aUrl)</code>• 설명 : EmailAttachment 객체에 첨부할 파일이 위치한 URL을 설정하거나 설정된 값을 받아오는 메소드다. 위의 get/setPath 메소드와 같은 역할을 하지만 경로를 지정하는 방식에 따른 매개변수의 타입이 다르다.



4. 첨부 파일 이메일

메일 보내기

보내는 사람: 박문석

받는 사람:

제 목:

파 일 첨 부: 첨부파일

메일 보내기

4. 첨부 파일 이메일

1.MailView의 생성자 수정

```
public class MailView extends JFrame {  
    public MailView(){  
        setTitle("메일 보내기");  
        setBounds(100,100,400,600);  
        setDefaultCloseOperation(EXIT_ON_CLOSE);  
  
        JPanel northPanel = new JPanel();  
        northPanel.setLayout(new GridLayout(5,2));  
        JLabel lblSend = new JLabel("보내는 사람: ", JLabel.RIGHT);  
        JLabel sendEmail = new JLabel("박문석");  
  
        JLabel lblReceive = new JLabel("받는 사람: ",  
JLabel.RIGHT);  
        JTextField receiveEmail = new JTextField(30);  
  
        JLabel lblTitle = new JLabel("제목: ",  
JLabel.RIGHT);  
        JTextField txtTitle = new JTextField(30);
```

4. 첨부 파일 이메일

```
JLabel lblFile = new JLabel("파 일 첨 부: ",  
JLabel.RIGHT);
```

```
 JButton btnFile = new JButton("첨부파일");
```

```
 JTextArea txtFiles = new JTextArea(2, 30);  
 JScrollPane sp1 = new JScrollPane(txtFiles);
```

```
 northPanel.add(lblSend);  
 northPanel.add(sendEmail);  
 northPanel.add(lblReceive);  
 northPanel.add(receiveEmail);  
 northPanel.add(lblTitle);  
 northPanel.add(txtTitle);  
 northPanel.add(lblFile);  
 northPanel.add(btnFile);  
 northPanel.add(new JLabel());  
 northPanel.add(sp1);
```

```
 add("North", northPanel);
```


4. 첨부 파일 이메일

```
TextArea content = new TextArea(20,20);  
ScrollPane sp = new JScrollPane(content);  
add(sp);
```

```
Panel southPanel = new Panel();  
Button btnSend = new Button("메일 보내기");  
southPanel.add(btnSend);  
add("South", btnSend);
```

```
EventHandler handler = new EventHandler(receiveEmail,  
txtTitle, content, txtFiles);  
btnSend.addActionListener(handler);  
btnFile.addActionListener(handler);  
setVisible(true);  
}  
}
```



4. 첨부 파일 이메일

2.EventHandler 클래스 수정

```
public class EventHandler implements ActionListener {  
    private JTextField txtReceiveEmail;  
    private JTextField txtTitle;  
    private JTextArea txtContent;  
    private JTextArea txtFiles;  
  
    public EventHandler(JTextField txtReceiveEmail, JTextField  
txtTitle, JTextArea txtContent, JTextArea txtFiles) {  
        super();  
        this.txtReceiveEmail = txtReceiveEmail;  
        this.txtTitle = txtTitle;  
        this.txtContent = txtContent;  
        this.txtFiles = txtFiles;  
    }  
}
```



4. 첨부 파일 이메일

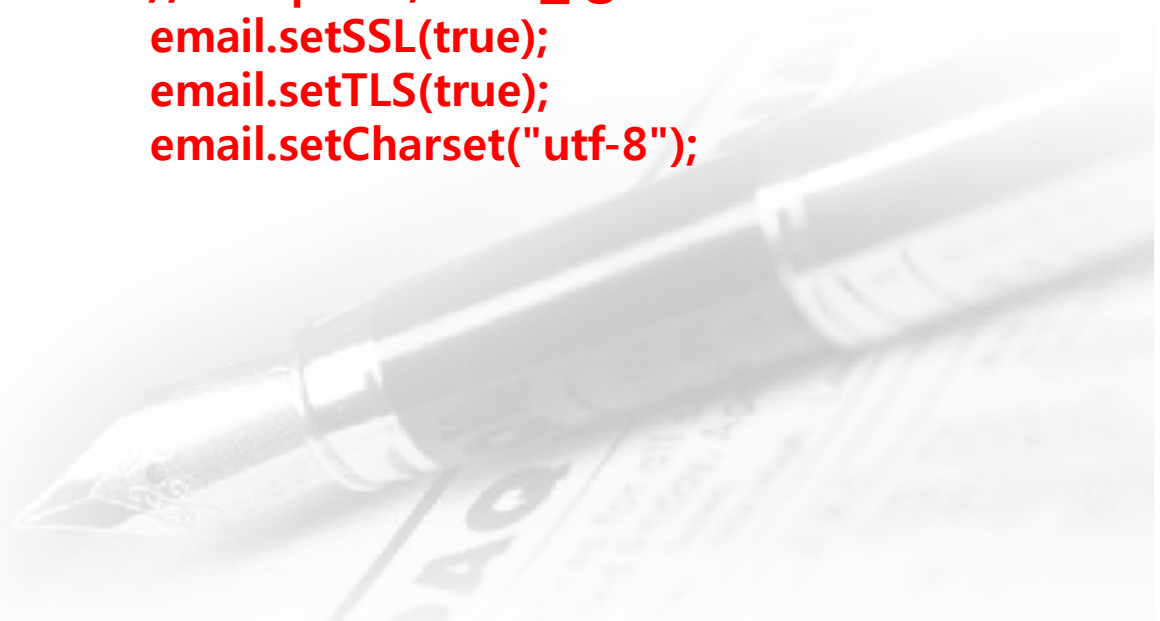
```
@Override
public void actionPerformed(ActionEvent e) {
    if (e.getActionCommand().equals("메일 보내기")) {
        String receiveEmail =
txtReceiveEmail.getText().trim();
        if (receiveEmail.length() == 0) {
            JOptionPane.showMessageDialog(null,
"받는 분의 이메일 주소가 없습니다.");
            return;
        }
        String title = txtTitle.getText().trim();
        if (title.length() == 0) {
            title = "무제";
        }
        String content = txtContent.getText().trim();
        if (content.length() == 0) {
            content = "냉무";
        }
    }
}
```

4. 첨부 파일 이메일

```
MultiPartEmail email = new MultiPartEmail();

// Smtplib 서버 연결 설정.
email.setHostName("smtp.naver.com");
email.setSmtpPort(587);
email.setAuthentication("ggangpae3",
    "wnddkd");

String rt = "";
try {
    // Smtplib SSL, TLS 설정
    email.setSSL(true);
    email.setTLS(true);
    email.setCharset("utf-8");
```



4. 첨부 파일 이메일

```
txtFiles.getText().split(",");

attach = new EmailAttachment();

if (txtFiles.getText().length() != 0) {
    String[] fileNames =

    for (String file : fileNames) {
        EmailAttachment

        attach.setName("");
        attach.setPath(file);
        email.attach(attach);
    }
}

// 받는 사람 설정
email.addTo(receiveEmail, "받는 이",

"utf-8");

// 보내는 사람 설정

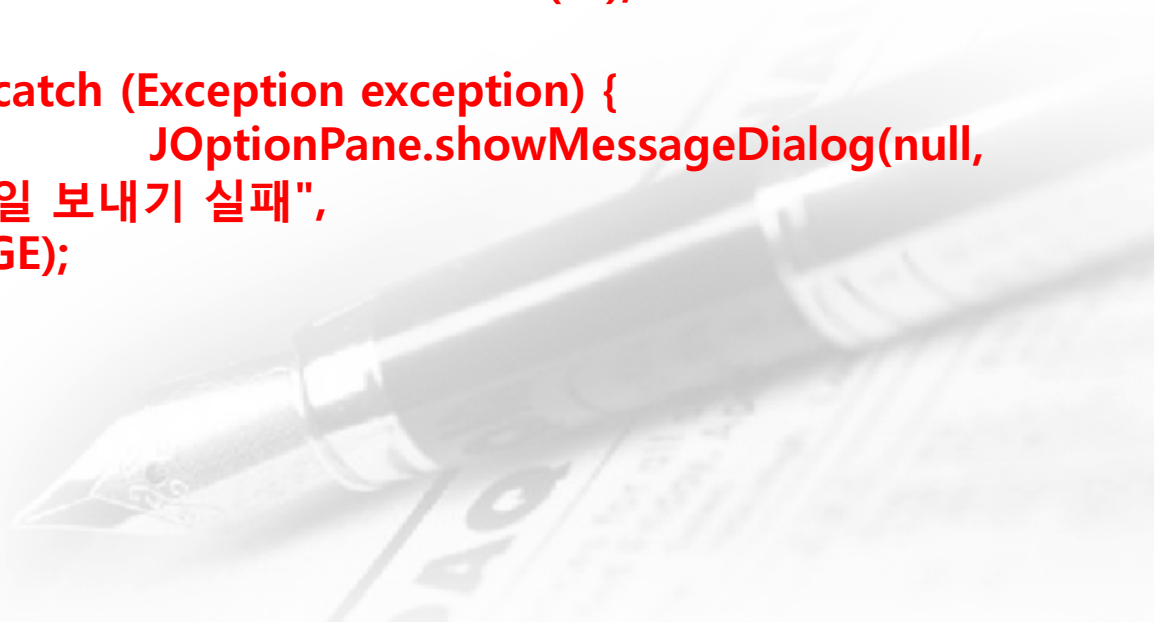
email.setFrom("ggangpae3@naver.com", "박문석", "utf-8");
```

4. 첨부 파일 이메일

```
// 제목 설정
email.setSubject(title);
// 본문 설정
email.setMsg(content);
email.send();
JOptionPane.showMessageDialog(null,
"메일 보내기 성공");


txtReceiveEmail.setText("");
txtTitle.setText("");
txtContent.setText("");

} catch (Exception exception) {
    JOptionPane.showMessageDialog(null,
exception.getMessage(), "메일 보내기 실패",
JOptionPane.ERROR_MESSAGE);
}
}
```



4. 첨부 파일 이메일

```
else {  
    JFileChooser fc = new JFileChooser();  
    fc.setMultiSelectionEnabled(true);  
    int result = fc.showOpenDialog(null);  
    if (result == JFileChooser.APPROVE_OPTION) {  
        File[] f = fc.getSelectedFiles();  
        for (File file : f) {  
            if(txtFiles.getText().trim().length() == 0) {  
                txtFiles.setText(file.getPath());  
            } else {  
                String origin =  
txtFiles.getText();  
                txtFiles.setText(origin + "," + file.getPath());  
            }  
        }  
    }  
}
```



5. HTML 이메일

▶ HTML 형식의 HtmlEmail 클래스

- ▶ **HtmlEmail 클래스** : 이메일을 HTML 형식으로 전송하고자 할 때 사용
- ▶ MuliPartEmail 클래스를 상속받고 있으므로 첨부 파일 전송까지 가능
- ▶ HtmlEmail 클래스는 기본 생성자만 제공



5. HTML 이메일

▶ HTML 형식의 HtmlEmail 클래스

메소드	설명
embed	<ul style="list-style-type: none">• 선언부 : <code>public String embed(File file)</code> <code>public String embed(File file, String cid)</code> <code>public String embed(String urlString, String name)</code> <code>public String embed(URL url, String name)</code> <code>public String embed(DataSource dataSource, String name)</code> <code>public String embed(DataSource dataSource, String name, String cid)</code>• 설명 : Email 본문에 이미지 등의 파일을 포함시킬 때 사용하는 메소드다. 이미지 파일의 유형에 따라서 여러 매개변수들을 받을 수 있도록 오버라이딩되어 있다.
setHtmlMsg	<ul style="list-style-type: none">• 선언부 : <code>public HtmlEmail setHtmlMsg(String aHtml)</code>• 설명 : 메일 본문에 HTML 태그를 포함한 메시지를 남기기 위해서 사용한다. 앞서 살펴본 <code>setMsg</code>과 같은 역할을 하지만 HTML 태그를 포함할 때 에러를 방지하는 역할을 한다.
setTextMsg	<ul style="list-style-type: none">• 선언부 : <code>public HtmlEmail setTextMsg(String aText)</code>• 설명 : HTML 태그가 포함되지 않는 순수 텍스트 메시지를 메일 본문에 포함하기 위해서 사용하는 메소드다.

5. HTML 이메일


```
import java.io.File;

import org.apache.commons.mail.EmailException;
import org.apache.commons.mail.HtmlEmail;

public class Main {
    public static void main(String[] args) {
        HtmlEmail htmlEmail = new HtmlEmail();

        // Smtп 서버 연결 설정.
        htmlEmail.setHostName("smtp.naver.com");
        htmlEmail.setSmtпPort(587);
        htmlEmail.setAuthentication("ggangpae3", "wnddkd");
        // 이미지 파일 생성
        File logoFile = new File("c://image1.png");

        String rt = "Failure";
```



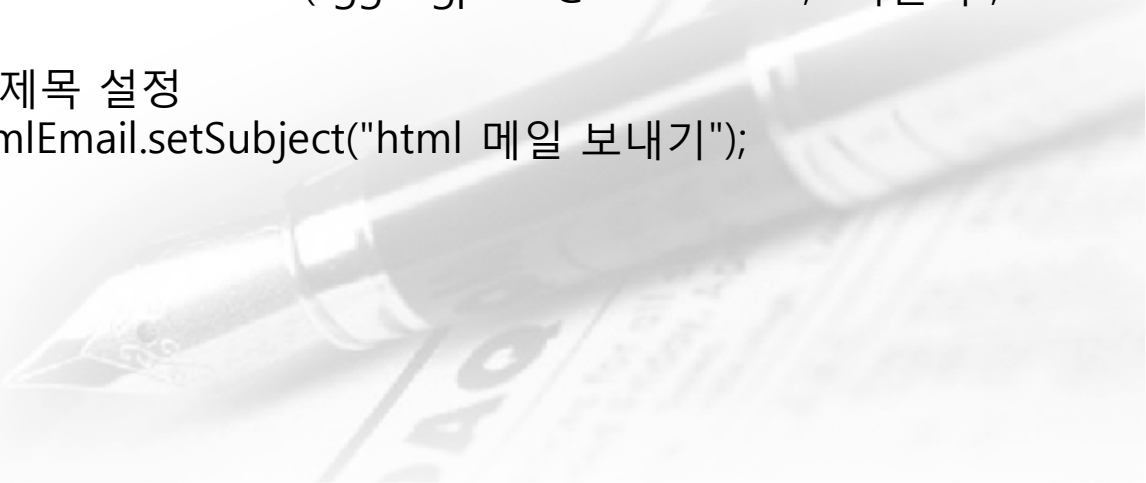
5. HTML 이메일

```
try
{
    // Smtп SSL, TLS 설정
    htmlEmail.setSSL(true);
    htmlEmail.setTLS(true);
    htmlEmail.setCharset("utf-8");

    // 받는 사람 설정
    htmlEmail.addTo("ggangpae1@naver.com", "박문석",
"utf-8");

    // 보내는 사람 설정
    htmlEmail.setFrom("ggangpae3@naver.com", "박문석",
"utf-8");

    // 제목 설정
    htmlEmail.setSubject("html 메일 보내기");
}
```



5. HTML 이메일

```
// HTML 태그가 포함된 메일 본문을 StringBuffer 객체를 사용하여 처리
StringBuffer sb = new StringBuffer();
sb.append("<html> <body>");
sb.append("안녕하세요 박문석 입니다. <br />");
sb.append("<img
src=cid:>").append(htmlEmail.embed(logoFile)).append(">");
sb.append("</body> </html>");
htmlEmail.setHtmlMsg(sb.toString());

rt = htmlEmail.send();
}
catch (EmailException e)
{
    System.out.println(e.getMessage());
}
}
```