목 차

Ⅰ. 오즈 엔터프라이즈 서버 API

- Class Cache
- Class ConnectionPool
- Class DataBind
- Class Log
- Class Module
- Class Monitor
- Class Repository
- Class Service
- Class Viewer

오즈 엔터프라이즈 서버에 관련된 각종 정보 조회와 실시간 환경 설정 변경 기능을 사용자 애플리케이션에서 직접 제어할 수 있도록 자바 API를 제공합니다.

다음은 오즈 엔터프라이즈 서버 API로 제공하는 주요 클래스에 대한 설명입니다.

클래스 이름	설명
Cache	오즈 서버 캐시 매니저와 관련된 처리를 수행합니다. 캐시 매니저 환경 설정 및 캐시 항목 삭제 등을 처리합니다.
Connection Pool	오즈 서버의 데이터베이스 연결 Pool에 대한 인터페이스로 해당 데이터베이스의 JDBC/ODBC 정보의 등록, 삭제 등의 관리 기능을 수행합니다.
DataBind	데이터 바인딩 모듈 관련 기능을 수행합니다.
Log	오즈 서버 로그 매니저와 관련된 처리를 수행하며 서버 운용시 발생되는 각종 메시지, 정보, 에러 등을 기록하기 위한 환경 설정을 변경합니다.
Module	오즈 데이터 모듈 관련 기능을 수행합니다.
Monitor	오즈 서버의 실시간 모니터링 관련 기능을 수행합니다.
Repository	오즈 서버에 탑재된 리파지토리의 관리 기능을 수행합니다.
Service	서버 정보, 상태, 처리, 메모리 수집 등의 처리 기능을 수행합니다.
Viewer	보고서 폼의 데이터 모듈, 뷰어 실행 등의 처리 기능을 수행합니다.

오즈 엔터프라이즈 서버 API를 사용하기 위해서는 다음과 같은 라이브러리 파일을 클래스 패스에 추가해 주어야 합니다.

라이브러리 파일명	설명
ozsfw35.jar	오즈 서버 및 Scheduler server에 접속하기 위한 파일입니다.
log4.jar	Server 내부의 클래스들을 사용할 때 Log를 남기기 위한 라이브러리 파일입니다. (API를 이용한 프로그램 실행시 classpath에 "log4.jar" 파일이 위치하여야 함)

Class Cache

Constructor Summary

- **Cache(String ip, int port, String id, String pw, boolean bAutoLogin, boolean useUSL)**
- **Cache(String url, String id, String pw, boolean bAutoLogin, boolean useUSL)**

Method Summary

- **SortProperties getCacheConfiguration()**
- **void setCacheConfiguration(SortProperties p)**

Constructor Detail

Prototype

//Daemon 타입 - 오즈 서버 타입이 TCP Server인 경우

```
public Cache(String ip, int port, String id, String pw,
boolean bAutoLogin, boolean useUSL)
```

//Servlet 타입 - 오즈 서버 타입이 HTTP Server인 경우

```
public Cache(String url, String id, String pw, boolean
bAutoLogin, boolean useUSL)
```

Argument

url	Servlet 타입 오즈 서버의 URL ex) String url = "http://127.0.0.1/oz/server";
ip	Daemon 타입 오즈 서버의 IP ex) String ip = "127.0.0.1";
port	Daemon 타입 오즈 서버의 포트 번호 ex) int port = 8003;
id	사용자 아이디 ex) String id = "admin";

	pw	사용자 패스워드 ex) String pw = "admin";
	bAutoLogin	자동 로그인 여부 ex) boolean bAutoLogin = true;
	useUSL	USL 사용 여부 ex) boolean useUSL = false;

Method Detail

■ **getCacheConfigration**

Prototype	`public SortProperties getCacheConfigration() throws OZCPException`
Definition	오즈 서버의 캐시 설정을 가져옵니다. 가져올 수 있는 설정값은 "SortProperties" 클래스의 key를 참조하시기 바랍니다.

■ **setCacheConfiguration**

Prototype	`public void setCacheConfiguration(SortProperties p) throws OZCPException`	
Definition	캐시 설정을 변경합니다. 변경할 수 있는 설정값은 "SortProperties" 클래스의 key를 참조하시기 바랍니다.	
Argument	*p*	캐시 설정 속성값

관련 Class

■ **OZCPException(oz.framework.cp.OZCPException)**

API를 사용할 때 생성되는 Exception입니다. 모든 API에서 OZCPException을 발생시킵니다.

- 메소드
  - getMessage

Prototype	`public String getMessage()`
Definition	에러 내용을 가져옵니다.

▪ getErrorCode

Prototype	`public int getErrorCode()`
Definition	에러 코드를 가져옵니다.

■ **SortProperties(oz.util.SortProperties.java)**

getCacheConfigration(), setCacheConfiguration()의 설정값을 반환하거나 설정하는데 사용되는 클래스입니다.

- 메소드

▪ getproperty

Prototype	`public synchronized String getProperty(String key)`
Definition	해당 key에 해당하는 속성값을 가져옵니다.

▪ setproperty

Prototype	`public synchronized Object setProperty(String key, String value)`
Definition	해당 key에 해당하는 속성을 지정한 값(value)으로 설정합니다.

- Key

getProperty()와 setProperty()에서 사용되는 key는 다음 표와 같습니다.

Key	Value	설명
Active	"true" "false"	캐시키 사용 여부 선택 ex) p.setProperty("datamodule.active", "false");
CACHE_FILE_PATH	저장 경로	캐시 저장 경로명 설정 ex) p.setProperty("CACHE_FILE_PATH","%OZ_HOME%/cache");
DM_CACHE_FILE_PATH	저장 경로	Data Module 캐시 저장 경로 ex) p.setProperty("DM_CACHE_FILE_PATH","%OZ_HOME%/cache_dm/");

memoryCacheValidTime	시간	메모리 캐시 타임 아웃(단위:초) ex) p.setProperty("datamodule.memoryCacheValidTime", "100");
diskCacheValidTime	시간	디스크 캐시 타임 아웃(단위:초) ex) p.setProperty("datamodule.diskCacheValidTime","100");
FreeMemoryPercentage	프리메모리	프리 메모리 퍼센트 설정 ex) p.setProperty("datamodule.freeMemoryPercentage", "20");

※ 참고사항 : 캐시키의 설정방법은 "오즈 엔터프라이즈 서버 관리자 매뉴얼의 Ⅳ.오즈 엔터프라이즈 서버 설정-cachemngr.properties"를 참조하시기 바랍니다.

Sample : CacheSample.java

```
package sample;

import oz.framework.api.Cache;
import org.apache.log4j.*;

import oz.util.SortProperties;

public class CacheSample {
   public static void main(String[] args) {
      //아래 라인을 주석 처리하면 사용자가 지정하지 않은 로그가 나타나지 않습니다.
       BasicConfigurator.configure();

      // OZServer Info.
      //*
      // Daemon
      String IP   = "127.0.0.1"; //서버가 설치되어 있는 호스트 컴퓨터의 IP
      int PORT = 8003;    //서버가 사용하는 TCP 포트
      /*/
      // Servlet
      String URL  = "http://www.oz.com/oz/server"; //Servlet 에 접근 가능한 URL
      /**/
      // User Info.
      String ID   = "admin"; //default
      String PWD  = "admin"; //default
```

```
        Cache cache = null;
        try {
            //*
            // Daemon
            cache = new Cache(IP, PORT, ID, PWD, false, false);
            /*/
            // Servlet
            cache = new Cache(URL, ID, PWD, false, false);
            /**/

            SortProperties p = new SortProperties();

            /*/
            // 캐시 환경설정 정보 저장 (setCacheConfiguration)
            p.setProperty("CACHE_FILE_PATH", "%OZ_HOME%/cache");
            // 디스크 저장 및 프리 메모리를 벗어났을때 저장 경로
            p.setProperty("DM_CACHE_FILE_PATH", "%OZ_HOME%/cache_dm");
            // 캐시 대상이 DataModule 일시 저장경로
            p.setProperty("datamodule.active", "true");
            //캐시매니저 사용 여부
            p.setProperty("datamodule.memoryCacheValidTime", "1000");
            //메모리 캐시 타임 아웃 설정
            p.setProperty("datamodule.diskCacheValidTime", "1000");
            //디스크 캐시 타임 아웃 설정
            p.setProperty("datamodule.freeMemoryPercentage", "21");
            //프리 메모리 퍼센트

            cache.setCacheConfiguration(p);

            /*/
            // 캐시 환경설정 정보 가져오기(getCacheConfiguration)
            p = cache.getCacheConfigration();
            /**/
            p.list(System.out);
        }
        catch(Exception e) {
            e.printStackTrace();
        }
    }
}
```

Class ConnectionPool

Constructor Summary

- **ConnectionPool(String ip, int port, String id, String pw, boolean bAutoLogin, boolean useUSL)**
- **ConnectionPool(String url, String id, String pw, boolean bAutoLogin, boolean useUSL)**

Method Summary

- **void addPool(ConnectionPoolInfo pool)**
- **void removePool(String pool)**
- **ConnectionPoolInfo[] getPoolInfoList()**
- **ConnectionPoolStatus[] getPoolStatusList()**
- **ConnectionPoolInfo getPoolInfo(String alias)**
- **void save()**

Constructor Detail

Prototype	**//Daemon 타입 - 오즈 서버 타입이 TCP Server인 경우** `public ConnectionPool(String ip, int port, String id, String pw, boolean bAutoLogin, boolean useUSL)` **//Servlet 타입 - 오즈 서버 타입이 HTTP Server인 경우** `public ConnectionPool(String url, String id, String pw, boolean bAutoLogin, boolean useUSL)`	
Argument	*url*	Servlet 타입 오즈 서버의 URL ex) String url = "http://127.0.0.1/oz/server";
	ip	Daemon 타입 오즈 서버의 IP ex) String ip = "127.0.0.1";

	port	Daemon 타입 오즈 서버의 포트 번호 ex) int port = 8003;
	id	사용자 아이디 ex) String id = "admin";
	pw	사용자 패스워드 ex) String pw = "admin";
	bAutoLogin	자동 로그인 여부 ex) boolean bAutoLogin = true;
	useUSL	USL 사용 여부 ex) boolean useUSL = false;

Method Detail

■ addPool

Prototype	`public void addPool(ConnectionPoolInfo pool) throws OZCPException`
Definition	ConnectionPool을 추가합니다. 추가할 ConnectionPool은 "ConnectionPoolInfo" 클래스에서 선언되어진 멤버 변수에 값을 설정하여 추가합니다.
Argument	*pool* 추가할 ConnectionPool의 정보인 ConnectionPoolInfo 객체

■ removePool

Prototype	`public void removePool(String pool) throws OZCPException`
Definition	해당 이름의 ConnectionPool을 삭제합니다.
Argument	*pool* ConnectionPool 이름

■ getPoolInfoList

Prototype	`public ConnectionPoolInfo[] getPoolInfoList() throws OZCPException`
Definition	모든 ConnectionPool에 대한 ConnectionPoolInfo 리스트를 가져옵니다.

■ getPoolStatusList

Prototype	`public ConnectionPoolStatus[] getPoolStatusList() throws OZCPException`

Definition	모든 ConnectionPool에 대한 상태 정보를 가져옵니다.

■ **getPoolInfo**

Prototype	`public ConnectionPoolInfo getPoolInfo(String alias) throws OZCPException`
Definition	해당 ConnectionPool에 대한 ConnectionPoolInfo 리스트를 가져옵니다.
Argument	*alias* ConnectionPool의 앨리어스 이름

■ **save**

Prototype	`public void save() throws OZCPException`
Definition	서버에 모든 ConnectionPool의 정보를 저장합니다.

관련 Class

■ **ConnectionPoolInfo(oz.framework.db.ConnectionPoolInfo.class)**

데이터베이스 연결에 대한 각종 정보들을 가지고 있는 클래스입니다.

■ **ConnectionPoolStatus(oz.framework.db.ConnectionPoolStatus.class)**

ConnectionPool의 상태 정보를 멤버 변수로 가지고 있는 클래스입니다.

- 멤버 변수
  - public final static int OK = 1;

    ConnectionPool의 현재 상태를 나타냅니다. 각 상태값에 대한 설명은 아래 표와 같습니다.

Status 값	설명
1	**OK** ConnectionPool 생성 성공
-1	**DRIVER_ERROR** ConnectionPool을 생성하는데 필요한 JDBC 드라이버를 불러오는데 실패
-2	**CONNECTION_ERROR** ConnectionPool을 생성 중 DBMS에서 초기 연결 획득 실패

  - public final static int DRIVER_ERROR = -1;
  - public final static int CONNECTION_ERROR = -2;

Sample : ConnectionPoolSample.java

```
package sample;

import java.util.HashMap;
import oz.framework.api.ConnectionPool;
import oz.framework.db.ConnectionPoolInfo;
import oz.framework.db.ConnectionPoolStatus;
import org.apache.log4j.*;

public class ConnectionPoolSample {
    public static void main(String[] args) {
        //아래 라인을 주석 처리하면 사용자가 지정하지 않은 로그가 나타나지 않습니다.
        BasicConfigurator.configure();

        // OZServer Info.

        //*
        // Daemon
        String IP = "127.0.0.1"; //서버가 설치되어 있는 호스트 컴퓨터의 IP
        int PORT = 8003; //서버가 사용하는 TCP 포트

        /*/
        // Servlet
        String URL = "http://www.oz.com/oz/server"; //Servlet 에 접근 가능한 URL

        /**/
        // User Info.
        String ID = "admin";//default
        String PWD = "admin";//default
        ConnectionPool conPool = null;

        try {
            //*
            // Daemon
            conPool = new ConnectionPool(IP, PORT, ID, PWD, false, false);

            /*/
            // Servlet
            conPool = new ConnectionPool(URL, ID, PWD, false, false);

            /**/
            // 커넥션풀 추가(addPool)
            ConnectionPoolInfo poolInfo = new ConnectionPoolInfo();
            poolInfo.setAlias("forcs"); //커넥션 풀 이름
            poolInfo.setVendor("mssql"); //데이터베이스 타입(현재 MSSQL)
```

```
            // 서버의 연결정보를 저장한다.
            // 각 벤더별 map 의 키는 dbconfig.xml 에 db 연결 jdbc url 부분을 참조
            HashMap infos = new HashMap();
            infos.put("serverAddress", "127.0.0.1");
            infos.put("portNo", "1433");
            infos.put("dbName", "master");
            infos.put("user", "sa");
            infos.put("password", "1588");
            poolInfo.setItems(infos);

            poolInfo.setMaxConns(20); //데이터베이스 최대 접속
            poolInfo.setInitConns(1); //데이터베이스 초기 접속
            conPool.addPool(poolInfo);

            // 커넥션풀 제거하기(removePool)
            String conPoolName = "forcs"; //제거할 커넥션 풀 이름
            conPool.removePool(conPoolName);
            // ConnectionPoolStatus 의 정보 리스트 가져오기(getPoolStatusList)
            ConnectionPoolStatus[]
poolStatusList=conPool.getPoolStatusList();
            for (int i = 0; i < poolStatusList.length; i++) {
               ConnectionPoolStatus cps = poolStatusList[i];

               //커넥션풀의 상태
               System.out.println(i);
               System.out.println("StatusString=" + cps.getStatusString());

               //프리 커넥션 풀
               System.out.println("free=" + new
Integer(cps.getFreeConnectionCount()));

               //체크아웃커넥션
               System.out.println("checkedout=" + new
Integer(cps.getCheckedOutConnectionCount()));
               System.out.println();
            }
            //커넥션풀 정보 저장하기(save)
            conPool.save();
        }
        catch(Exception e) {
               e.printStackTrace();
        }
    }
}
```

Class DataBind

Constructor Summary

- **DataBind(String ip, int port, String id, String pw, boolean bAutoLogin, boolean useUSL)**
- **DataBind(String url, String id, String pw, boolean bAutoLogin, boolean useUSL)**

Method Summary

- **void setDataBindConfigration(SortProperties config)**
- **SortProperties getDataBindConfigration()**

Constructor Detail

Prototype	**//Daemon 타입 - 오즈 서버 타입이 TCP Server인 경우** `public DataBind(String ip, int port, String id, String pw, boolean bAutoLogin, boolean useUSL)` **//Servlet 타입 - 오즈 서버 타입이 HTTP Server인 경우** `public DataBind(String url, String id, String pw, boolean bAutoLogin, boolean useUSL)`	
Argument	*url*	Servlet 타입 오즈 서버의 URL ex) String url = "http://127.0.0.1/oz/server";
	ip	Daemon 타입 오즈 서버의 IP ex) String ip = "127.0.0.1";
	port	Daemon 타입 오즈 서버의 포트 번호 ex) int port = 8003;
	id	사용자 아이디 ex) String id = "admin";

	pw	사용자 패스워드 ex) String pw = "admin";
	bAutoLogin	자동 로그인 여부 ex) boolean bAutoLogin = true;
	useUSL	USL 사용 여부 ex) boolean useUSL = false;

Method Detail

■ setDataBindConfigration

Prototype	`public void setDataBindConfigration(SortProperties config) throws OZCPException`
Definition	DataBind 설정 즉, "databind.properties"에 설정되어 있는 값을 변경할 수 있습니다.
Argument	*config* DataBind 설정 속성

■ getDataBindConfigration

Prototype	`public SortProperties getDataBindConfigration() throws OZCPException`
Definition	DataBind의 설정값 즉, "databind.properties"에 설정되어 있는 값을 가져옵니다.

- Key

  setDataBindConfigration()와 getDataBindConfigration()에서 사용되는 key는 다음 표와 같습니다.

Key	Value	설명
ConcurrentFetchSize	버퍼 크기	FetchType이 "Concurrent"일 때 클라이언트로 데이터를 전송할때 Stream 버퍼의 크기를 설정합니다. 단위는 byte, 기본값은 4096, 최소값은 256입니다. ※ 주의사항 : 음수를 설정하거나 잘못된 형식으로 설정하면 기본값으로 설정되고, 최소값보다 작은 양수로 설정하면 최소값으로 설정됩니다.

ConcurrentFirstRow	행 개수	FetchType이 "Concurrent"일 때 클라이언트로 데이터를 전송할 때 맨 처음 전송되는 첫 데이터의 행 개수를 설정합니다. 단위는 개수, 기본값은 0입니다. ※ 주의사항 : 0보다 작은 값을 설정한 경우에는 기본값으로 설정됩니다.

Sample : DataBindSample.java

```
package sample;

import oz.framework.api.DataBind;
import org.apache.log4j.*;
import oz.util.SortProperties;

public class DataBindSample {
   public static void main(String[] args) {
      //아래 라인을 주석 처리하면 사용자가 지정하지 않은 로그가 나타나지 않습니다.
      BasicConfigurator.configure();
      // OZServer Info.
      String IP = "127.0.0.1"; //서버가 설치되어 있는 호스트 컴퓨터의 IP
      int PORT = 8003; //서버가 사용하는 TCP 포트
      // User Info.
      String ID = "admin"; //default
      String PWD = "admin"; //default

      DataBind dataBind = null;
      try {
         dataBind = new DataBind(IP, PORT, ID, PWD, false, false);
         SortProperties p = new SortProperties();

         // 데이터바인드 환경설정 정보 저장 (setDataBindConfiguration)
         //Stream 버퍼의 크기를 설정
         p.setProperty("ConcurrentFetchSize", "4096");

         //최초 row 갯수 지정
         p.setProperty("ConcurrentFirstRow", "0");
         dataBind.setDataBindConfiguration(p);

         // 데이터바인드 환경설정 정보 가져오기(getDataBindConfiguration)
         p = dataBind.getDataBindConfigration();
         java.util.Vector vec = p.propertyNames();
         for(int i=0; i<vec.size(); i++) {
            String name = (String)vec.elementAt(i);
```

```
            System.out.println(name + "=" + p.getProperty(name));
         }
      }
      catch (Exception e) {
         e.printStackTrace();
      }
   }
}
```

Class Log

Constructor Summary

- **Log(String ip, int port, String id, String pw, boolean bAutoLogin, boolean useUSL)**
- **Log(String url, String id, String pw, boolean bAutoLogin, boolean useUSL)**

Method Summary

- **String getConfiguration()**
- **byte[] downloadLog()**
- **void downloadLog(String fileName)**
- **void setConfiguration(String logs)**
- **void setPriority(String p)**

Constructor Detail

Prototype	//Daemon 타입 - 오즈 서버 타입이 TCP Server인 경우 public Log(String ip, int port, String id, String pw, boolean bAutoLogin, boolean useUSL) //Servlet 타입 - 오즈 서버 타입이 HTTP Server인 경우 public Log(String url, String id, String pw, boolean bAutoLogin, boolean useUSL)	
Argument	*url*	Servlet 타입 오즈 서버의 URL ex) String url = "http://127.0.0.1/oz/server";
	ip	Daemon 타입 오즈 서버의 IP ex) String ip = "127.0.0.1";
	port	Daemon 타입 오즈 서버의 포트 번호 ex) int port = 8003;

	id	사용자 아이디 ex) String id = "admin";
	pw	사용자 패스워드 ex) String pw = "admin";
	bAutoLogin	자동 로그인 여부 ex) boolean bAutoLogin = true;
	useUSL	USL 사용 여부 ex) boolean useUSL = false;

Method Detail

■ **getConfigure**

Prototype	`public String getConfigure() throws OZCPException`
Definition	로그 설정을 서버로부터 가져옵니다.

■ **downloadLog**

Prototype	`public byte[] downloadLog() throws OZCPException`
Definition	서버로부터 로그 파일을 다운로드합니다.

■ **downloadLog**

Prototype	`public void downloadLog(String fileName) throws OZCPException, IOException`	
Definition	서버로부터 다운로드한 로그 파일을 로컬에 저장합니다.	
Argument	*fileName*	저장할 파일명

■ **setConfigure**

Prototype	`public void setConfigure(String logs) throws OZCPException`	
Definition	로그 환경 설정값을 변경합니다.	
Argument	*logs*	변경할 로그 환경 설정값, "key=value" 형식으로 설정 ex) String logs="Priority=DEBUG" ex) String logs="CONSOLE.Layout=%r[%t]%p%c{1}%X-%m%n

■ **setPriority**

Prototype	`public void setPriority(String p) throws OZCPException`
Definition	로그의 레벨을 설정합니다.(INFO, DEBUG, ERROR)
Argument	*p* 변경할 로그 레벨값

Sample : LogSample.java

```
package sample;

import oz.framework.api.Log;
import org.apache.log4j.*;

public class LogSample {
   public static void main(String[] args) {
      //아래 라인을 주석 처리하면 사용자가 지정하지 않은 로그가 나타나지 않습니다.
      BasicConfigurator.configure();
      // OZServer Info.
      // Daemon
      String IP = "127.0.0.1"; //서버가 설치되어 있는 호스트 컴퓨터의 IP
      int PORT = 8003; //서버가 사용하는 TCP 포트
      // Servlet
      String URL  = "http://www.oz.com/oz/server"; //Servlet에 접근 가능한 URL
      // User Info.
      String ID = "admin"; //default
      String PWD = "admin"; //default
      Log log = null;
      try {
         // Daemon
         log = new Log(IP, PORT, ID, PWD, false, false);
         // Servlet
         log = new Log(URL, ID, PWD, false, false);
         // 로그 환경 정보 가져오기(getConfigure)
         String conf = null;
         conf = log.getConfigure();
         System.out.println(conf);
         // 로그 환경 설정(setConfigure, setProirity)
         String logs = "Priority=INFO";
         //log.setConfigure(logs);
         log.setPriority("DEBUG");
         // 로그파일 내려받기(downloadLog)
         byte b[] = log.downloadLog();
         String fileName = "server.log";
         // 내용을 파일로 저장합니다.
         log.downloadLog(fileName);
```

```
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
    }
}
```

Class Module

Constructor Summary

- Module(String ip, int port, String id, String pw, boolean bAutoLogin, boolean useUSL)
- Module(String url, String id, String pw, boolean bAutoLogin, boolean useUSL)

Method Summary

- InputString getOZD(String item, String category, String[] urls)
- InputStream getOZD(String item, String category, Hashtable formparam, Hashtable odiparam, boolean memoallowed, String password, String id, String pw, HttpServletRequest request)
- InputStream getOZD(String item, String category, Hashtable formparam, Hashtable odiparam, Hashtable odipath , boolean memoallowed, String password, String id, String pwd, HttpServletRequest request)
- InputStream getOZD(String item, String category, String serverDMType, Hashtable formparam, Hashtable odiparam, Hashtable odipath, boolean memoallowed, String password, String id, String pwd, HttpServletRequest request)
- InputStream getOZU(String item, String category, String[] urls)
- void addODIParameter(String odiName, String key, String value)
- void addODIParameter(String odiName, String item, String category, Hashtable paramHash)
- void addParameter(String key, String value)
- void addApplicationParameter(String key, String value)
- void registODIPath (String odiName, String path)
- void saveOZD(String fileName, String item, String category, String[] urls)
- void saveOZU(String fileName, String item, String category, String[] urls)

Constructor Detail

Prototype	//Daemon 타입 - 오즈 서버 타입이 TCP Server인 경우 public Module(String ip, int port, String id, String pw, boolean bAutoLogin, boolean useUSL) //Servlet 타입 - 오즈 서버 타입이 HTTP Server인 경우 public Module(String url, String id, String pw, boolean bAutoLogin, boolean useUSL)	
Argument	*url*	Servlet 타입 오즈 서버의 URL ex) String url = "http://127.0.0.1/oz/server";
	ip	Daemon 타입 오즈 서버의 IP ex) String ip = "127.0.0.1";
	port	Daemon 타입 오즈 서버의 포트 번호 ex) int port = 8003;
	id	사용자 아이디 ex) String id = "admin";
	pw	사용자 패스워드 ex) String pw = "admin";
	bAutoLogin	자동 로그인 여부 ex) boolean bAutoLogin = true;
	useUSL	USL 사용 여부 ex) boolean useUSL = false;

Method Detail

■ **getOZD**

Prototype	public InputStream getOZD(String item, String category, String[] urls) throws OZCPException
Definition	서버로부터 보고서 파일과 SDM 파일을 가져와 OZD 파일을 생성하여 가져옵니다. OZD 파일을 생성할 때 urls에 지정된 이미지 파일도 함께 저장합니다. ※ 주의사항 : API 함수를 이용하여 데이터를 바인딩할 경우에는 항상 DM_TYPE="Momory", FetchType="Batch"로 바인딩됩니다. 대용량 데이터를 바인딩할 경우에는 메모리를 많이 사용하거나 결과물을 얻는데 시간이 걸릴 수 있습니다.

Argument	*item*	아이템 이름 (보고서 파일인 OZR 파일 이름)
	category	카테고리 이름
	urls	OZD에 함께 저장할 이미지 파일의 URL

Prototype	`public final InputStream getOZD(String item, String category, Hashtable formparam, Hashtable odiparam, boolean memoallowed, String password, String id, String pwd, HttpServletRequest request) throws Exception`
Definition	오즈 서블릿 서버를 상속받아 OZD를 생성하는 오즈 서블릿 서버를 생성하고, 새로 생성된 오즈 서블릿 서버와 통신하여 OZD 파일을 만듭니다. 데몬 타입의 서버에서는 사용될 수 없고, 서블릿 타입의 서버에서만 사용됩니다. ※ 참고사항 : 함수에 대한 좀 더 자세한 사용 방법은 아래 샘플 중 "RequestOZDSample.java"를 참조하시기 바랍니다. ※ 주의사항 : API 함수를 이용하여 데이터를 바인딩할 경우에는 항상 DM_TYPE="MEMORY", FetchType="BATCH"로 바인딩됩니다. 대용량 데이터를 바인딩할 경우에는 메모리를 많이 사용하거나 결과물을 얻는데 시간이 걸릴 수 있습니다.
Argument	*item* : 아이템 이름 (보고서 파일인 OZR 파일 이름) *category* : 카테고리 이름 *formparam* : 폼 패러미터 ※ 주의사항 : 폼 패러미터는 Hashtable에 패러미터명(String), 패러미터값(String등 패러미터의 데이터형) 형태로 입력하여야 합니다. *odiparam* : ODI 패미터터 ※ 주의사항 : ODI 패러미터는 Hashtable에 ODI명(String), 패러미터값(Hashtable) 형태로 입력하여야 하며, 패러미터값은 다시 Hashtable 형태로 패러미터명(String), 패러미터값(String) 형태로 입력하여야 합니다. *memoallowed* : OZD 문서에 메모 혀용 여부 설정 *password* : OZD 문서를 열 때 적용할 패스워드 *id* : 오즈 서버에 접속하기 위한 사용자 아이디 *pw* : 오즈 서버에 접속하기 위한 사용자 패스워드 *request* : HttpServletRequest

Prototype	`public final InputStream getOZD(String item, String category, String serverDMType, Hashtable formparam, Hashtable odiparam, Hashtable odiPath, boolean memoallowed, String password, String id, String pwd, HttpServletRequest`

	request)	
Definition	오즈 서블릿 서버를 상속받아 OZD를 생성하는 오즈 서블릿 서버를 생성하고, 새로 생성된 오즈 서블릿 서버와 통신하여 OZD 파일을 만듭니다. 데몬 타입의 서버에서는 사용될 수 없고, 서블릿 타입의 서버에서만 사용됩니다. ※ 참고사항 : 함수에 대한 좀 더 자세한 사용 방법은 아래 샘플 중 "RequestOZDSample.java"를 참조하시기 바랍니다. ※ 주의사항 : API 함수를 이용하여 데이터를 바인딩할 경우에는 항상 DM_TYPE="MEMORY", FetchType="BATCH"로 바인딩됩니다. 대용량 데이터를 바인딩할 경우에는 메모리를 많이 사용하거나 결과물을 얻는데 시간이 걸릴 수 있습니다.	
Argument	*item*	아이템 이름 (보고서 파일인 OZR 파일 이름)
	category	카테고리 이름
	formparam	폼 패러미터 ※ 주의사항 : 폼 패러미터는 Hashtable에 패러미터명(String), 패러미터값(String등 패러미터의 데이터형) 형태로 입력하여야 합니다.
	odiparam	ODI 패미터터 ※ 주의사항 : ODI 패러미터는 Hashtable에 ODI명(String), 패러미터값(Hashtable) 형태로 입력하여야 하며, 패러미터값은 다시 Hashtable 형태로 패러미터명(String), 패러미터값(String) 형태로 입력하여야 합니다.
	odipath	ODI 파일 경로 (OZD 생성시 동적으로 변경할 ODI 파일 경로)
	memoallowed	OZD 문서에 메모 혀용 여부 설정
	pwd	OZD 문서를 열 때 적용할 패스워드
	id	오즈 서버에 접속하기 위한 사용자 아이디
	pw	오즈 서버에 접속하기 위한 사용자 패스워드
	request	HttpServletRequest

■ getOZU

Prototype	`public InputStream getOZU(String item, String category, String[] urls) throws OZCPException`

Definition	서버로부터 애플리케이션 파일과 SDM 파일을 가져와 OZU 파일을 생성하여 가져옵니다. ※ 주의사항 : API 함수를 이용하여 데이터를 바인딩할 경우에는 항상 DM_TYPE="Momory", FetchType="Batch"로 바인딩됩니다. 대용량 데이터를 바인딩할 경우에는 메모리를 많이 사용하거나 결과물을 얻는데 시간이 걸릴 수 있습니다. ※ 주의사항 : "FetchUnit"은 반드시 "DM_PER_DATAMODULE"로 설정하여야 합니다.	
Argument	*item*	아이템 이름 (애플리케이션 파일인 OZA 파일 이름)
	category	카테고리 이름
	urls	OZU에 함께 저장할 이미지 파일의 URL

■ **addODIParameter**

Prototype	`public void addODIParameter(String odiName, String key, String value)`	
Definition	SDM 파일을 만들기 위해 ODI에 지정되어 있는 ODI 패러미터 값을 설정합니다. ODI 패러미터 값을 설정하지 않을 경우에는 ODI 기본 패러미터 기본값이 사용됩니다.	
Argument	*odiName*	ODI 이름
	key	ODI 패러미터 이름
	value	ODI 패러미터 값

■ **addODIParameter**

Prototype	`public void addODIParameter(String odiName, String item, String category, Hashtable paramHash) throws IllegalArgumentException`	
Definition	SDM 파일을 만들기 위해 ODI에 지정되어 있는 ODI 패러미터 값을 설정합니다. ODI 패러미터 값을 설정하지 않을 경우에는 ODI 기본 패러미터 기본값이 사용됩니다. ODI 패러미터 값에 따라 각각 다른 SDM 파일이 필요할 경우 각각의 패러미터 값 별로 SDM 생성 단위를 만들고자 할 때 사용합니다.	
Argument	*odiName*	디자이너에서 지정한 ODI 이름
	item	ODI 파일명
	category	ODI 파일 경로
	paramHash	패러미터 Key, Value 가 설정되어 있는 Hashtable

※ 주의사항 : OZU 생성시 paramHash 주의사항

OZU를 생성할 때 addODIParameter()의 paramHash값에 null을 넣을 경우에는 반드시 addApplicationParameter(Key,Value)에 ODI 패러미터를 설정해 주어야 합니다.

ex) addApplicationParameter를 이용한 패러미터 설정 예

```
module.addApplicationParameter("odi.odinames", "sample");
module.addApplicationParameter("odi.sample.pcount", "1");
module.addApplicationParameter("odi.sample.args1", "deptid=501");
```

■ **addParameter**

Prototype	`public void addParameter(String key, String value)`	
Definition	SDM 파일을 만들기 위해 폼에 지정되어 있는 폼 패러미터 값을 설정합니다. 폼 패러미터 값을 설정하지 않을 경우에는 폼 패러미터 기본값이 사용됩니다.	
Argument	*key*	폼 패러미터 이름
	value	폼 패러미터 값

■ **addApplicationParameter**

Prototype	`public void addApplicationParameter(String key, String value)`	
Definition	SDM 파일을 만들기 위해 애플리케이션의 패러미터 값을 설정합니다. 현재 애플리케이션에서 설정할 수 있는 패러미터는 ODI 패러미터만 있으므로 ODI에 설정되어 있는 ODI 패러미터 값을 설정하시기 바랍니다.	
Argument	*key*	ODI 패러미터 이름
	value	ODI 패러미터 값

■ **registODIPath**

Prototype	`public void registODIPath (String odiName, String path) throws IllegalArgumentException`	
Definition	OZD 생성시 동적으로 변경할 ODI파일을 등록합니다.	
Argument	*odiName*	ODI 이름
	path	ODI 파일 경로

■ **saveOZD**

Prototype	`public void saveOZD(String fileName, String item, String category, String[] urls) throws OZCPException`

Definition	보고서 파일을 OZD 파일로 저장합니다. ※ 주의사항 : API 함수를 이용하여 데이터를 바인딩할 경우에는 항상 DM_TYPE="Momory", FetchType="Batch"으로 바인딩됩니다. 대용량 데이터를 바인딩할 경우에는 메모리를 많이 사용하거나 결과물을 얻는데 시간이 걸릴 수 있습니다.	
Argument	*fileName*	저장 경로를 포함한 OZD 파일명 설정한 저장 폴더가 없을 경우에는 자동으로 폴더 생성함
	item	보고서 파일(.ozr) 이름
	category	보고서 파일(.ozr)의 카테고리 이름
	Urls	OZD에 함께 저장할 이미지 파일의 URL

■ **saveOZU**

Prototype	`public void saveOZU(String filename, String item, String category, String[] urls) throws OZCPException`	
Definition	애플리케이션 파일을 OZU 파일로 저장합니다. ※ 주의사항 : API 함수를 이용하여 데이터를 바인딩할 경우에는 항상 DM_TYPE="Momory", FetchType="Batch"으로 바인딩됩니다. 대용량 데이터를 바인딩할 경우에는 메모리를 많이 사용하거나 결과물을 얻는데 시간이 걸릴 수 있습니다. ※ 주의사항 : "FetchUnit"은 반드시 "DM_PER_DATAMODULE"로 설정하여야 합니다.	
Argument	*fileName*	저장 경로를 포함한 OZU 파일명 설정한 저장 폴더가 없을 경우에는 자동으로 폴더 생성함
	item	애플리케이션 파일(.oza) 이름
	category	애플리케이션 파일(.oza)의 카테고리 이름
	Urls	OZU에 함께 저장할 이미지 파일의 URL

Sample : ModuleSample.java

```
package sample;

import java.io.*;
import oz.framework.api.Module;
import org.apache.log4j.*;

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

```
        //아래 라인을 주석 처리하면 사용자가 지정하지 않은 로그가 나타나지 않습니다.
        BasicConfigurator.configure();
        // OZServer Info.
        //*
        // Daemon
        String IP = "127.0.0.1"; //서버가 설치되어 있는 호스트 컴퓨터의 IP
        int PORT = 8003; //서버가 사용하는 TCP 포트
        /*/
        // Servlet
        String URL = "http://www.oz.com/oz/server";
        /**/
        // User Info.
        String ID = "admin"; //default
        String PWD = "admin"; //default
        Module module = null;
        InputStream stream = null;
        try {
           //*
           // Daemon
           module = new Module(IP, PORT, ID, PWD, false, false);
           /*/
           // Servlet
           module = new Module(URL, ID, PWD, false, false);
           /**/
           //module.addODIParameter("대용량 100 만", "rowcount", "40000");
           //replace ODI
           //module.registODIPath("대용량 100 만 ", "/대용량테스트/ 대용량 100
           //만.odi");
           // module.addParameter("server", "127.0.0.1");
           // module.addParameter("port", "8003");
           // module.addParameter("reportname", "/대용량/대용량 100 만.ozr");
           // 기본설정으로 ozd 만들 경우
           //stream = module.getOZD("대용량 100 만.ozr", "/대용량");
           //FileOutputStream out = new FileOutputStream("D:/TEST_OZD.ozd");
           // saveOZD(file name, item name, category name)
           //module.saveOZD("D:/TEST_OZD.ozd", "대용량 100 만.ozr", "/대용량" );
           //out.flush();
           //out.close();
           // ---------------------------------------------------------------
           // OZD 생성시 이미지 같이 저장
           // set form parameters
           module.addParameter("FORM_PARAM1", "COCO 폼 파라미터 1");
           module.addParameter("FORM_PARAM2", "COCO 폼 파라미터 2");
           // set odi parameters
           module.addODIParameter("OZP_PARAM", "ODI_PARAM1", "테스트 ODI 파라미터
1");
           module.addODIParameter("OZP_PARAM", "ODI_PARAM2", "테스트 ODI 파라미터
2");
```

```
            module.addODIParameter("OZP_DB", "condition", "KIA");
            String[] urls = {
               "http://211.116.251.20/img/ban_center.gif",
               "http://211.116.251.20/img/img_interview.gif",
               "ozp://img/netiq.gif",     // ozp:// image
               "ozp://img/credos.gif",    // ozp:// image - table
               "ozp://img/enterprise.gif", // ozp:// image - table
               "ozp://img/sephia.gif"     // ozp:// image - table
            };
            module.setPassword("1234");
            module.setMemoAllowed(true);
            module.saveOZD("D:/OZP_TEST01.ozd", "OZP_TEST.ozr", "/", urls);
         }
         catch(Exception e) {
            e.printStackTrace();
         }
      }
}
```

Sample : RequestOZDSample.java

```
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.ServletConfig;
import oz.server.OZServlet;
import javax.servlet.ServletException;

import java.io.*;
import java.util.*;

import oz.framework.cp.io.OZDataOutputStream;

public class RequestOZDSample extends OZServlet {
   private static final int PROTOCOL_VER = 2005;
   private static final String _ROOT_PATH = "D:/";
   private byte[] _buf = new byte[1024];
   public void init(ServletConfig config) throws ServletException {
      super.init(config);
   }

   public void doGet(HttpServletRequest request,
      HttpServletResponse response) throws ServletException, IOException
   {
      doPost(request, response);
   }
```

```
public void doPost(HttpServletRequest request,
   HttpServletResponse response) throws ServletException, IOException
{
   try {
      System.out.println("Start to get OZD");
      long _JOB_ID = System.currentTimeMillis();
      String tempOZDFile = _ROOT_PATH + "sample" + _JOB_ID + ".ozd";
      String tempWMFile  = _ROOT_PATH + "sample" + _JOB_ID + ".mtx";
      // Make OZD
      makeOZD(tempOZDFile, request);
   }
   catch(Exception e) {
      ByteArrayOutputStream bout = new ByteArrayOutputStream();
      e.printStackTrace(new PrintStream(bout));
      bout.flush();
      byte[] b = bout.toByteArray();
      String error = new String(bout.toByteArray());
      bout.close();
      System.out.println(error);
      PrintWriter writer = response.getWriter();
      writer.println("RequestOZDSample : Error");
      writer.println();
      writer.println(error);
      writer.flush();
   }
}

private void writeFile(String filename, OZDataOutputStream out)
   throws IOException
{
   BufferedInputStream bin = null;
   try {
      ByteArrayOutputStream bout = new ByteArrayOutputStream();
      int len;
      bin = new BufferedInputStream(new FileInputStream(filename));
      while((len = bin.read(_buf)) >= 0) {
         bout.write(_buf, 0, len);
      }
      bout.flush();
      byte[] buf = bout.toByteArray();
      bout.close();
      out.writeInt(buf.length);
      out.write(buf,0,buf.length);
   }
   finally {
      if(bin != null) {
         try {
```

```
                bin.close();
            }
            catch(Exception ex) {
            }
        }
    }
}

private void makeOZD(String filename, HttpServletRequest request)
    throws Exception
{
    InputStream stream = null;
    try {
        // Info.
        String   _ITEM = "parameter_test.ozr";
        String   _CATEGORY = "/";
        boolean  _MEMOALLOW = true;
        String   _PASSWORD = "1234";
        String   _UID = "admin";
        String   _PWD = "admin";

        //Form parameter
        Hashtable _FORM_PARAM = new Hashtable();
        _FORM_PARAM.put("formparam1", "FORM 1");
        _FORM_PARAM.put("formparam2", "FORM 2");
       //ODI parameter
        Hashtable _ODI_PARAM = new Hashtable();
        Hashtable _odi_param = new Hashtable();
        _odi_param.put("odiparam1", "ODI 1");
        _odi_param.put("odiparam2", "ODI 2");
        _ODI_PARAM.put("parameter_test", _odi_param);
        // call make ozd
        stream = getOZD(_ITEM, _CATEGORY,
                    _FORM_PARAM, _ODI_PARAM, _MEMOALLOW, _PASSWORD,
                    _UID, _PWD, request);
        //ODI parameter
        //Hashtable _ODI_PARAM = new Hashtable();
        //Hashtable _ODI_PARAM = new Hashtable();
        //Hashtable _ODI_PATH = new Hashtable();
        //_ODI _PARAM.put("odiparam1", "ODI 1");
        //_ODI _PARAM.put("odiparam2", "ODI 2");
        //_ODI_PARAM.put("parameter_test", _odi_param);
        //_ODI_PATH.put("parameter_test", "/test/parameter_test.odi");
        //call make ozd
        //stream = getOZD(_ITEM, _CATEGORY,
        //          _FORM_PARAM, _ODI_PARAM, _ODI_PATH, _MEMOALLOW,
        //          _PASSWORD, _UID, _PWD, request);
        FileOutputStream out = new FileOutputStream(filename);
```

```
            copy(stream, out);
            out.flush();
            out.close();
        }
        catch(Exception e) {
            e.printStackTrace();
            throw e;
        }
        finally {
            if(stream != null) {
                try {
                    stream.close();
                }
                catch(Exception e) {}
            }
        }
    }

    // Util method
    public static int copy(InputStream is, OutputStream os) throws IOException
    {
        byte[] buf = new byte[1024];
        int rt = 0;
        int len;
        while((len = is.read(buf)) >= 0) {
            os.write(buf, 0, len);
            rt += len;
        }
        return rt;
    }
}
```

Sample : ModuleOZUSample.java

```
package sample;

import java.io.*;
import oz.framework.api.Module;
import org.apache.log4j.*;
import java.util.Hashtable;

public class ModuleOZUSample{
  public static void main(String[] args){
    //아래 라인을 주석 처리하면 사용자가 지정하지 않은 로그가 나타나지 않습니다.
    BasicConfigurator.configure();
```

```
  // OZServer Info.
  String IP = "127.0.0.1"; //서버가 설치되어 있는 호스트 컴퓨터의 IP
  int PORT = 8003; //서버가 사용하는 TCP 포트
  // User Info.
  String ID = "admin"; //default
  String PWD = "admin"; //default
  Module module = null;
  InputStream stream = null;
  try{
    module = new Module(IP, PORT, ID, PWD, false, /*usl*/ false);
    //ozu를 생성하기 위해서는 반드시 설정해야 할 파라메터
    module.addApplicationParameter("odi.fetchunit", "DM_PER_DATAMODULE");
    module.addApplicationParameter("odi.odinames", "sample");
    module.addApplicationParameter("odi.sample.pcount", "1");
    module.addApplicationParameter("odi.sample.args1", "deptid=501");
    Hashtable hash = new Hashtable();
    hash.put("deptid", "501");
    String[] urls = {
       "http://www.anykiki.com/custom/casting/일반/Dana/미.gif"};
    // paramHash에 NULL이 들어가면 위의 addApplication Parameter에 해쉬에
    // 들어갈 내용을 반드시 설정을 해줘야 한다.
    module.addODIParameter("sample", "sample.odi", "/sample", null);
    module.addODIParameter("sample", "sample.odi", "/sample", hash);
    //ozu 저장할 파일이름, oza 파일이름, 카테고리, 해당 URL
    module.saveOZU("D:/ozu.ozu", "sample.oza", "/sample", urls);
  }
  catch(Exception e){
     e.printStackTrace();
  }
 }
}
```

Class Monitor

Constructor Summary

- **Monitor(String ip, int port, String id, String pw, boolean bAutoLogin, boolean useUSL)**
- **Monitor (String url, String id, String pw, boolean bAutoLogin, boolean useUSL)**

Method Summary

- **Versions getVersions()**
- **MemoryStatus getMemoryInfo()**
- **byte[] downloadMonitorLog()**
- **void downloadMonitorLog(String fileName)**

Constructor Detail

Prototype	//Daemon 타입 - 오즈 서버 타입이 TCP Server인 경우 `public Monitor(String ip, int port, String id, String pw, boolean bAutoLogin, boolean useUSL)` //Servlet 타입 - 오즈 서버 타입이 HTTP Server인 경우 `public Monitor(String url, String id, String pw, boolean bAutoLogin, boolean useUSL)`	
Argument	*url*	Servlet 타입 오즈 서버의 URL ex) String url = "http://127.0.0.1/oz/server";
	ip	Daemon 타입 오즈 서버의 IP ex) String ip = "127.0.0.1";
	port	Daemon 타입 오즈 서버의 포트 번호 ex) int port = 8003;

	id	사용자 아이디 ex) String id = "admin";
	pw	사용자 패스워드 ex) String pw = "admin";
	bAutoLogin	자동 로그인 여부 ex) boolean bAutoLogin = true;
	useUSL	USL 사용 여부 ex) boolean useUSL = false;

Method Detail

■ **getVersions**

Prototype	`public Versions getVersions() throws OZCPException`
Definition	서버의 버전 정보와 각 모듈별 버전 정보를 가져옵니다.

■ **getMemoryInfo**

Prototype	`public MemoryStatus getMemoryInfo() throws OZCPException`
Definition	현재의 메모리 사용량(전체 메모리, 사용 메모리, 비사용 메모리) 정보를 가져옵니다.

■ **downloadMonitorLog**

rototype	`public byte[] downloadMonitorLog() throws OZCPException`
Definition	서버에 저장된 모니터 로그 파일을 가져옵니다.

■ **downloadMonitorLog(String fileName)**

Prototype	`public void downloadMonitorLog(String fileName) throws OZCPException, IOException`
Definition	서버에 저장된 모니터 로그 파일을 가져와서 fileName으로 지정된 위치에 로그 파일을 저장합니다.

관련 Class

■ **MemoryStatus(oz.server.monitor.MemoryStatus)**

Server가 실행되고 있는 System의 메모리 상태를 나타냅니다.

■ **Versions(oz.server.monitor.Versions)**

Server와 Server가 실행되고 있는 System의 버전 정보들을 나타냅니다.

- 멤버 변수
  - public String osName : Server가 실행되고 있는 OS의 이름
  - public String osVersion : Server가 실행되고 있는 OS의 버전
  - public String javaVendor : Server가 실행되고 있는 JVM의 제작 회사
  - public String javaVersion : Server가 실행되고 있는 JVM의 Version
  - public String OZServerVersion : 오즈 서버의 버전
  - public String CPRelease : OZ Common Protocol의 릴리즈 정보
  - public int CPProtocol : OZ Common Protocol의 프로토콜 버전
  - public String DMRelease : OZ Data Module의 릴리즈 정보
  - public int DMStreaming : OZ Data Module의 Streaming 프로토콜 버전

Sample : MonitorSample.java

```
package sample;

import oz.framework.api.Monitor;
import oz.server.monitor.Versions;
import oz.server.monitor.MemoryStatus;
import org.apache.log4j.*;

public class MonitorSample {
   public static void main(String[] args) {
      //아래 라인을 주석 처리하면 사용자가 지정하지 않은 로그가 나타나지 않습니다.
       BasicConfigurator.configure();

       // OZServer Info.
       //*
       // Daemon
       String IP = "127.0.0.1"; //서버가 설치되어 있는 호스트 컴퓨터의 IP
       int PORT = 8003; //서버가 사용하는 TCP 포트
```

```
        /*/
        // Servlet
        String URL  = "http://www.oz.com/oz/server"; //Servlet 에 접근 가능한 URL
        /**/
        // User Info.
        String ID = "admin"; //default
        String PWD = "admin"; //default

        Monitor monitor = null;
        try {
            //*
            // Daemon
            monitor = new Monitor(IP, PORT, ID, PWD, false, false);
            /*/
            // Servlet
            monitor = new Monitor(URL, ID, PWD, false, false);
            /**/

            // 서버와 모듈별 버전 정보(getServerInformation)
           Versions v = monitor.getVersions();
           v._printOut();

           // 메모리 사용량 확인(getServerStatus)
           MemoryStatus ms = monitor.getMemoryInfo();

           // 사용메모리 = 전체메모리 - 비사용메모리
           long usedMemory = ms.getTotalMemory() - ms.getFreeMememoy();

           // 가져온 서버 메모리 사용량 상태값
           System.out.println("Total Memory="+ms.getTotalMemory());
           System.out.println("Used Memory="+usedMemory);
           System.out.println("Free Memory="+ms.getFreeMememoy());
           System.out.println("");

           // 모니터 로그를 내려 받습니다 .
           byte [] logBytes = monitor.downloadMonitorLog();

           // 모니터 로그를 지정하는 파일로 저장합니다.
           String logFileName = "monitor.log";
           monitor.downloadMonitorLog(logFileName);

        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
    }
}
```

Class Repository

Constructor Summary

- Repository(String ip, int port, String id, String pw, boolean bAutoLogin, boolean useUSL)
- Repository(String url, String id, String pw, boolean bAutoLogin, boolean useUSL)

Method Summary

/ / Configuration

- public void setRepositoryConfig(SortProperties prop)
- public SortProperties getRepositoryConfig()

/ / User

- public int createUser(String userName, String pwd, int gid, String description)
- public void deleteUser(int uid)
- public void updateUserName(int uid, String userName)
- public String getUserNameByID(int uid)

/ / UserLogin

- public void disableUserLogin(String userName)
- public void updateLoginDefault(int loginDVal)
- public void enableUserLogin(String userName)
- public void userLogout(int uid)
- public boolean isUserLogin(int uid)
- public boolean loginToServer()

/ / UserPwd

- public boolean checkUserPwd(int uid, String pwd)

- public void updateUserPwd(int uid, String pwd)

// UserDesc

- public void updateUserDescription(int uid, String description)
- public String getUserDescription(int uid)

// UserID

- public int getGroupIdOfUser(int uid)
- public int getUserIdbyName(String userName)
- public void updateGroupIdOfUser(int gid, int uid)

// UserList

- public OZRepositoryUser[] getUserList()
- public OZRepositoryUser[] getUserListInGroup(int gid)
- public OZRepositoryUser[] getUserListAuthToItem(int itemid, byte perm)
- public OZRepositoryUser[] getUserListAuthToCategory(int categoryId, byte perm)

// Group

- public int createGroup(String groupName, int upperGid)
- public void deleteGroup(int gid)
- public void updateUpperGroupId(int gid, int upperGid)
- public void updateGroupName(int groupId, String groupName)

// GroupAdmin

- public void updateUserGroupAdmin(int uid, int gid)
- public boolean isUserGroupAdmin(int uid, int gid)

// GroupList

- public OZRepositoryGroup[] getGroupListInGroup(int gid)
- public OZRepositoryGroup getGroupInfo(int gid)
- public OZRepositoryGroup[] getGroupListAuthToItem(int itemid, byte perm)
- public OZRepositoryGroup[] getGroupListAuthToCategory(int categoryId, byte perm)

// Item

- public int createItem(String itemName, int itemType, String itemDescription, int cid, InputStream itemIn)
- public int createItem(String itemName, int itemType, String itemDescription, String categoryName, InputStream itemIn)
- public int getItemId(String itemName, int itemType, int cid)
- public int getItemId(String itemName, int itemType, String cName)
- public void deleteItem(int itemid)
- public InputStream getItem(int itemid)
- public void updateItemName(int itemId, String itemName)
- public InputStream getDirectItem(String itemName, int itemType, String categoryName)
- public InputStream getDirectItem(String itemName, int itemType, String categoryName, boolean compressedItem)
- public void updateItem(int itemId, InputStream input)
- public void updateDirectItem(String itemName, int itemType, String categoryName, InputStream input)
- public boolean hasItemInRepository(String itemName, int itemType, String categoryName)

// InfoByItem

- public int getCategoryIdOfItem(int itemid)
- public void updateCategoryIdOfItem(int cid, int itemid)

// ItemList

- public OZRepositoryItem[] getItemList()
- public OZRepositoryItem getItemInfo(int itemid)
- public OZRepositoryItem[] getItemListInCategory(int cid)
- public OZRepositoryItem[] getItemListInCategory(String cName)
- public OZRepositoryItem[] getItemListInCategory(int cid, int uid, byte perm)
- public OZRepositoryItem[] getItemListInCategory(String categoryName, int uid, byte perm)
- public OZRepositoryItem[] getItemListInCategoryAuthGroup(int cid, int gid, byte perm)

- public OZRepositoryItem[] getItemListInCategoryAuthGroup(String categoryName, int gid, byte perm)
- public OZRepositoryItem[] getItemListAuthToUser(int uid, byte perm)
- public OZRepositoryItem[] getItemListAuthToGroup(int gid, byte perm)

// Category

- public int createCategory(String categoryName, int upperCid)
- public int createCategory(String categoryPath)
- public void deleteCategory(int cid)
- public int getCategoryID(String fullPath)
- public void updateUpperCategoryId(int cid, int upperCid)
- public void updateCategoryName(int cid, String categoryName)
- public int getItemCountInCategory(int cid)
- public OZRepositoryCategory[] getCategoryListInCategory(int cid)
- public OZRepositoryCategory[] getCategoryListInCategory(int cid, int uid, byte perm)
- public OZRepositoryCategory[] getCategoryListInCategoryAuthGroup(int cid, int gid, byte perm)
- public OZRepositoryCategory getCategoryInfo(int cid)
- public OZRepositoryCategory[] getCategoryListAuthToUser(int uid, int cid, byte perm)
- public OZRepositoryCategory[] getCategoryListAuthToGroup(int gid, int cid, byte perm)

// CheckInOut

- public void checkOutItem(int itemId, int uid, String checkoutFolder)
- public void undoCheckOutItem(int itemId, int uid)
- public void checkInItem(boolean keepChkOut, int itemId, int uid, InputStream itemIn)
- public boolean isCheckOutUser(int itemId, int uid)

// History

- public void rollBackItem(int itemId, int itemVersion)
- public InputStream getSpecifiedVersionItem(int itemId, int itemVersion)
- public OZRepositoryHistory[] getHistoryItemList(int itemId)

■ **public void clearHistoryItem(int itemId, int itemVersion)**

Constructor Detail

Prototype	//Daemon 타입 - 오즈 서버 타입이 TCP Server인 경우 public Repository(String ip, int port, String id, String pw, boolean bAutoLogin, boolean useUSL) //Servlet 타입 - 오즈 서버 타입이 HTTP Server인 경우 public Repository(String url, String id, String pw, boolean bAutoLogin, boolean useUSL)	
Argument	*url*	Servlet 타입 오즈 서버의 URL ex) String url = "http://127.0.0.1/oz/server";
	ip	Daemon 타입 오즈 서버의 IP ex) String ip = "127.0.0.1";
	port	Daemon 타입 오즈 서버의 포트 번호 ex) int port = 8003;
	id	사용자 아이디 ex) String id = "admin";
	pw	사용자 패스워드 ex) String pw = "admin";
	bAutoLogin	자동 로그인 여부 ex) boolean bAutoLogin = true;
	useUSL	USL 사용 여부 ex) boolean useUSL = false;

Method Detail

// Configuration

■ **setRepositoryConfig**

Prototype	public void setRepositoryConfig(SortProperties prop) throws OZCPException

Definition	리파지토리 설정을 변경합니다. "repository.properties"에 설정되어 있는 값을 변경할 수 있으며, 설정할 수 있는 Key와 값은 아래 표를 참조하시기 바랍니다.
Argument	*prop* 리파지토리 설정 속성

■ **getRepositoryConfig**

Prototype	`public SortProperties getRepositoryConfig() throws OZCPException`
Definition	리파지토리의 설정값을 가져옵니다. "repository.properties"에 설정되어 있는 값을 가져올 수 있으며, 가져올 수 있는 Key와 값은 아래 표를 참조하시기 바랍니다.

- Key

  setRepositoryConfig()와 getRepositoryConfig()에서 사용되는 key는 다음 표와 같습니다.

Key	Value	설명
REPOSITORY_TYPE	"RDB" "BUILTIN"	리파지토리 타입 ex) prop.setProperty("REPOSITORY_TYPE","RDB");
REPOSITORY_FILE_PATH	저장 경로	리파지토리 파일 경로 ex) prop.setProperty("REPOSITORY_FILE_PATH","c:/temp_repository");
REPOSITORY_ITEM_NUMBER_PER_DIRECTORY	아이템 파일수	디렉토리내 저장 가능한 아이템 파일 개수 (기본값 : "500") ex) prop.setProperty("REPOSITORY_ITEM_NUMBER_PER_DIRECTORY","100");
REPOSITORY_HISTORY_ITEM_VALID_DAYS	일 수	아이템 유효 기간 ex) prop.setProperty("REPOSITORY_HISTORY_ITEM_VALID_DAYS","20");
REPOSITORY_ADD_COMPRESSED_ITEM	"true" "false"	아이템 압축 여부 ex) prop.setProperty("REPOSITORY_ADD_COMPRESSED_ITEM","false");

/ / User

■ **createUser**

Prototype	public int createUser(String userName, String pwd, int gid, String description) throws OZCPException	
Definition	새로운 사용자를 생성하고, 생성된 사용자의 ID를 반환합니다.	
Argument	*userName*	사용자 이름
	pwd	패스워드
	gid	그룹 ID
	description	사용자에 대한 설명 내용

■ **deleteUser**

Prototype	public void deleteUser(int uid) throws OZCPException	
Definition	지정한 사용자 ID에 해당하는 사용자의 모든 정보를 삭제합니다.	
Argument	*uid*	삭제할 사용자의 ID

■ **updateUserName**

Prototype	public void updateUserName(int uid, String userName) throws OZCPException	
Definition	지정한 사용자 ID에 해당하는 사용자의 이름을 변경합니다.	
Argument	*uid*	사용자 ID
	userName	변경할 사용자 이름

■ **getUserNameByID**

Prototype	public String getUserNameByID(int uid) throws OZCPException	
Definition	지정한 사용자 ID에 해당하는 사용자의 이름을 가져옵니다.	
Argument	*uid*	사용자 ID

/ / UserLogin

■ **disableUserLogin**

Prototype	public void disableUserLogin(String userName) throws OZCPException	
Definition	지정한 사용자 이름에 해당하는 사용자를 로그인하지 못하도록 설정합니다.	
Argument	*userName*	로그인하지 못하도록 할 사용자 이름

■ **updateLoginDefault**

Prototype	public void updateLoginDefault(int loginDVal) throws OZCPException
Definition	지정한 사용자 ID에 해당하는 사용자로 기본 접속 사용자를 변경합니다.
Argument	*loginDVal* 기본 접속 사용자로 설정할 사용자 ID

■ **enableUserLogin**

Prototype	public void enableUserLogin(String userName) throws OZCPException
Definition	지정한 사용자 이름에 해당하는 사용자를 로그인 가능하도록 설정합니다.
Argument	*userName* 로그인 가능하도록 할 사용자 이름

■ **userLogout**

Prototype	public void userLogout(int uid) throws OZCPException
Definition	지정한 사용자 ID에 해당하는 사용자를 로그 아웃 즉, 접속 해제시킵니다.
Argument	*uid* 접속 해제시킬 사용자 ID

■ **isUserLogin**

Prototype	public boolean isUserLogin(int uid) throws OZCPException
Definition	지정한 사용자 ID에 해당하는 사용자가 로그인 상태 즉, 접속 상태인지 확인합니다.
Argument	*uid* 접속 상태를 확인할 사용자 ID

■ **loginToServer**

Prototype	public boolean loginToServer() throws OZCPException
Definition	현재 서버에 접속되어 있는지 여부를 확인합니다.

// UserPwd

■ **checkUserPwd**

Prototype	public boolean checkUserPwd(int uid, String pwd) throws OZCPException	
Definition	사용자의 패스워드가 맞는지 여부를 체크합니다.	
Argument	*uid*	패스워드를 체크할 사용자 ID
	pwd	체크할 패스워드

■ **updateUserPwd**

Prototype	public void updateUserPwd(int uid, String pwd) throws OZCPException	
Definition	지정한 사용자 ID에 해당하는 사용자의 패스워드를 변경합니다.	
Argument	*uid*	패스워드를 변경할 사용자 ID
	pwd	변경할 패스워드

// UserDesc

■ **updateUserDescription**

Prototype	public void updateUserDescription(int uid, String description) throws OZCPException	
Definition	지정한 사용자 ID에 해당하는 사용자의 사용자 설명 내용을 변경합니다.	
Argument	*uid*	사용자 설명 내용을 변경할 사용자 ID
	description	변경할 사용자 설명

■ **getUserDescription**

Prototype	public String getUserDescription(int uid) throws OZCPException	
Definition	지정한 사용자 ID에 해당하는 사용자의 사용자 설명 내용을 가져옵니다.	
Argument	*uid*	사용자 설명 내용을 가져올 사용자 ID

// UserID

■ **getGroupIdOfUser**

Prototype	public int getGroupIdOfUser(int uid) throws OZCPException	
Definition	지정한 사용자 ID에 해당하는 사용자의 그룹 ID를 가져옵니다.	
Argument	*uid*	그룹 ID를 가져올 사용자 ID

■ **getUserIdbyName**

Prototype	public int getUserIdbyName(String userName) throws OZCPException	
Definition	지정한 사용자 이름에 해당하는 사용자의 ID를 가져옵니다.	
Argument	*userName*	사용자 ID를 가져올 사용자 이름

■ **updateGroupIdOfUser**

Prototype	`public void updateGroupIdOfUser(int gid, int uid) throws OZCPException`	
Definition	사용자의 그룹 ID를 변경합니다.	
Argument	*gid*	변경할 그룹 ID
	uid	그룹 ID를 변경할 사용자 ID

/ / UserList

■ **getUserList**

Prototype	`public OZRepositoryUser[] getUserList() throws OZCPException`
Definition	리파지토리에 접근 가능한 모든 사용자의 정보를 가져옵니다.

■ **getUserListInGroup**

Prototype	`public OZRepositoryUser[] getUserListInGroup(int gid) throws OZCPException`	
Definition	지정한 그룹 ID에 등록되어 있는 모든 사용자의 정보를 가져옵니다.	
Argument	*gid*	사용자의 정보를 가져올 그룹 ID

■ **getUserListAuthToItem**

Prototype	`public OZRepositoryUser[] getUserListAuthToItem(int itemid, byte perm) throws OZCPException`	
Definition	지정한 아이템 ID에 대해 perm 이상의 권한을 가지고 있는 모든 사용자 정보를 가져옵니다.	
Argument	*itemId*	아이템 ID
	perm	권한 정보

■ **getUserListAuthToCategory**

Prototype	`public OZRepositoryUser[] getUserListAuthToCategory(int categoryId, byte perm) throws OZCPException`	
Definition	지정한 카테고리 ID에 대해 perm 이상의 권한을 가지고 있는 모든 사용자 정보를 가져옵니다.	
Argument	*categoryId*	카테고리 ID
	perm	권한 정보

/ / Group

■ **createGroup**

Prototype	`public int createGroup(String groupName, int upperGid) throws OZCPException`	
Definition	새로운 그룹을 생성하고, 생성된 그룹의 ID를 반환합니다.	
Argument	*groupName*	새로 생성할 그룹의 이름
	upperGId	새로 생성할 그룹의 상위 그룹 ID

■ **deleteGroup**

Prototype	`public void deleteGroup(int gid) throws OZCPException`	
Definition	지정한 그룹 ID에 해당하는 그룹을 삭제합니다.	
Argument	*gid*	삭제할 그룹 ID

■ **updateGroupName**

Prototype	`public void updateGroupName(int gid, String groupName) throws OZCPException`	
Definition	지정한 그룹 ID에 해당하는 그룹의 이름을 변경합니다.	
Argument	*gid*	그룹의 이름을 변경할 그룹 ID
	groupName	변경할 그룹의 이름

/ / GroupAdmin

■ **updateUserGroupAdmin**

Prototype	`public void updateUserGroupAdmin(int uid, int gid) throws OZCPException`	
Definition	지정한 그룹 ID에 해당하는 그룹의 관리자를 지정한 사용자로 변경합니다.	
Argument	*uid*	관리자를 변경할 그룹 ID
	gid	새로운 관리자로 설정할 사용자 ID

■ **isUserGroupAdmin**

Prototype	`public boolean isUserGroupAdmin(int uid, int gid) throws OZCPException`	
Definition	지정한 사용자 ID에 해당하는 사용자가 해당 그룹의 관리자인지 여부를 확인합니다.	
Argument	*uid*	관리자인지 체크할 사용자 ID

	gid	그룹 ID

/ / GroupList

■ **getGroupListInGroup**

Prototype	`public OZRepositoryGroup[] getGroupListInGroup(int gid) throws OZCPException`	
Definition	지정한 그룹 ID에 속한 모든 하위 그룹의 정보를 가져옵니다.	
Argument	*gid*	하위 그룹의 정보를 가져올 그룹 ID

■ **getGroupInfo**

Prototype	`public OZRepositoryGroup getGroupInfo(int gid) throws OZCPException`	
Definition	지정한 그룹 ID에 해당하는 그룹의 정보를 가져옵니다.	
Argument	*gid*	그룹 ID

■ **getGroupListAuthToItem**

Prototype	`public OZRepositoryGroup[] getGroupListAuthToItem(int itemid, byte perm) throws OZCPException`	
Definition	지정한 아이템 ID에 대해 perm 이상의 권한을 가지고 있는 모든 그룹 정보를 가져옵니다.	
Argument	*itemId*	아이템 ID
	perm	권한 정보

■ **getGroupListAuthToCategory**

Prototype	`public OZRepositoryGroup[] getGroupListAuthToCategory(int categoryId, byte perm) throws OZCPException`	
Definition	지정한 카테고리 ID에 대해 perm 이상의 권한을 가지고 있는 모든 그룹 정보를 가져옵니다.	
Argument	*categoryId*	카테고리 ID
	perm	권한 정보

■ **getSubGroupList**

Prototype	`public OZRepositoryGroup[] getSubGroupList(int gid) throws OZCPException`

Definition	해당 그룹의 서브 그룹 리스트를 가져옵니다. getGroupListInGroup()와의 차이점은 getSubGroupList()는 해당 그룹의 서브 그룹을 가져올 때 Recursive하게 가져옵니다. 즉, 서브 그룹의 서브 그룹 ...형태로 가져옵니다.	
Argument	*gid*	그룹의 이름을 변경할 그룹 ID
	groupName	변경할 그룹의 이름

// Item

■ createItem

Prototype	`public int createItem(String itemName, int itemType, String itemDescription, int cid, InputStream itemIn) throws OZCPException` `public int createItem(String itemName, int itemType, String itemDescription, String categoryName, InputStream itemIn) throws OZCPException`	
Definition	새로운 아이템을 생성하고, 생성된 아이템의 ID를 반환합니다.	
Argument	*itemName*	새로 생성할 아이템의 이름
	itemType	새로 생성할 아이템의 타입
	itemDescription	새로 생성할 아이템의 설명 내용
	cid	새로 생성할 아이템의 ID
	itemIn	새로 생성할 아이템의 입력 스트림
	categoryName	새로 생성할 아이템이 카테고리 이름

■ getItemId

Prototype	`public int getItemId(String itemName, int itemType, int cid) throws OZCPException` `public int getItemId(String itemName, int itemType, String cName) throws OZCPException`	
Definition	아이템 ID를 가져옵니다.	
Argument	*itemName*	가져올 아이템의 이름
	itemType	가져올 아이템의 타입
	cid	가져올 아이템의 카테고리 ID
	cName	가져올 아이템의 카테고리의 이름

■ **deleteItem**

Prototype	public void deleteItem(int itemId) throws OZCPException
Definition	지정한 아이템을 리파지토리에서 삭제합니다.
Argument	*itemId* 삭제할 아이템의 ID

■ **getItem**

Prototype	public InputStream getItem(int itemId) throws OZCPException
Definition	지정한 아이템 ID에 해당하는 아이템을 가져옵니다.
Argument	*itemId* 가져올 아이템의 ID

■ **updateItemName**

Prototype	public void updateItemName(int itemId, String itemName) throws OZCPException	
Definition	지정한 아이템 ID에 해당하는 아이템의 이름을 변경합니다.	
Argument	*itemId*	이름을 변경할 아이템의 ID
	itemName	변경할 아이템 이름

■ **getDirectItem**

Prototype	public InputStream getDirectItem(String itemName, int itemType, String categoryName) throws OZCPException public InputStream getDirectItem(String itemName, int itemType, String categoryName, boolean compressedItem) throws OZCPException	
Definition	아이템을 가져옵니다.	
Argument	*itemName*	가져올 아이템의 이름
	itemType	가져올 아이템의 타입
	categoryName	가져올 아이템의 카테고리 이름
	compressedItem	가져올 아이템의 압축 여부

■ **updateItem**

Prototype	public void updateItem(int itemId, InputStream input) throws OZCPException	
Definition	지정한 아이템 ID에 해당하는 아이템을 변경합니다.	
Argument	*itemId*	변경할 아이템의 ID

	input	변경할 내용

■ **updateDirectItem**

Prototype	public void updateDirectItem(String itemName, int itemType, String categoryName, InputStream input) throws OZCPException	
Definition	지정한 아이템 ID에 해당하는 아이템을 변경합니다.	
Argument	*itemName*	변경할 아이템의 이름
	itemType	변경할 아이템의 타입
	categoryName	변경할 아이템의 카테고리 이름
	input	변경할 내용

■ **hasItemInRepository**

Prototype	public boolean hasItemInRepository(String itemName, int itemType, String categoryName) throws OZCPException	
Definition	지정한 아이템이 해당 카테고리에 있는지 여부를 확인합니다.	
Argument	*itemName*	체크할 아이템의 이름
	itemType	체크할 아이템의 타입
	categoryName	체크할 카테고리의 이름

// InfoByItem

■ **getCategoryIdOfItem**

Prototype	public int getCategoryIdOfItem(int itemId) throws OZCPException	
Definition	지정한 아이템 ID에 해당하는 아이템이 속한 카테고리 ID를 가져옵니다.	
Argument	*itemId*	아이템 ID

■ **updateCategoryIdOfItem**

Prototype	public void updateCategoryIdOfItem(int cid, int itemId) throws OZCPException	
Definition	지정한 아이템 ID에 해당하는 아이템의 카테고리를 변경합니다.	
Argument	*cid*	변경할 카테고리 ID
	itemId	카테고리를 변경할 아이템 ID

// ItemList

■ getItemList

Prototype	`public OZRepositoryItem[] getItemList() throws OZCPException`
Definition	리파지토리 내 모든 아이템의 정보를 가져옵니다.

■ getItemInfo

Prototype	`public OZRepositoryItem getItemInfo(int itemId) throws OZCPException`
Definition	지정한 아이템 ID에 해당하는 아이템의 정보를 가져옵니다.
Argument	*itemId* 정보를 가져올 아이템의 ID

■ getItemListInCategory

Prototype	`public OZRepositoryItem[] getItemListInCategory(int cid) throws OZCPException` `public OZRepositoryItem[] getItemListInCategory(String categoryFullPath) throws OZCPException` `public OZRepositoryItem[] getItemListInCategory(int cid, int uid, byte perm) throws OZCPException` `public OZRepositoryItem[] getItemListInCategory(String categoryFullPath, int uid, byte perm) throws OZCPException`
Definition	지정한 옵션에 해당하는 모든 아이템의 정보를 가져옵니다.
Argument	*cid* 가져올 아이템의 카테고리 ID
	categoryFullPath 가져올 아이템의 카테고리 전체 경로
	uid 가져올 아이템의 사용자 ID
	perm 권한 정보

■ getItemListInCategoryAuthGroup

Prototype	`public OZRepositoryItem[] getItemListInCategoryAuthGroup(int cid, int gid, byte perm) throws OZCPException` `public OZRepositoryItem[] getItemListInCategoryAuthGroup (String categoryFullPath, int gid, byte perm) throws OZCPException`
Definition	지정한 옵션에 해당하는 아이템의 정보를 가져옵니다.

Argument	*cid*	가져올 아이템의 카테고리 ID
	gid	가져올 아이템의 그룹 ID
	perm	권한 정보
	categoryFullPath	가져올 아이템의 카테고리 전체 경로

■ **getItemListAuthToUser**

Prototype	`public OZRepositoryItem[] getItemListAuthToUser(int uid, byte perm) throws OZCPException`	
Definition	지정한 사용자 ID에 해당하는 사용자에게 perm 이상의 권한이 부여된 모든 아이템의 정보를 가져옵니다.	
Argument	*uid*	사용자 ID
	perm	권한 정보

■ **getItemListAuthToGroup**

Prototype	`public OZRepositoryItem[] getItemListAuthToGroup(int gid, byte perm) throws OZCPException`	
Definition	지정한 그룹 ID에 해당하는 그룹에게 perm 이상의 권한이 부여된 모든 아이템의 정보를 가져옵니다.	
Argument	*gid*	그룹 ID
	perm	권한 정보

/ / Category

■ **createCategory**

Prototype	`public int createCategory(String categoryName, int upperCid) throws OZCPException`	
Definition	카테고리를 새로 생성하고, 생성된 카테고리의 ID를 반환합니다.	
Argument	*categoryName*	생성할 카테고리의 이름
	upperCid	생성할 카테고리의 상위 카테고리 ID

■ **createCategory**

Prototype	`public int createCategory(String categoryPath) throws OZCPException`
Definition	지정된 경로로 카테고리를 새로 생성하고, 생성된 카테고리의 ID를 반환합니다.

Argument	*categoryPath*	새로 생성할 카테고리의 전체 경로

■ **deleteCategory**

Prototype	`public void deleteCategory(int cid) throws OZCPException`
Definition	지정한 카테고리 ID에 해당하는 카테고리를 삭제합니다.
Argument	*cid* 삭제할 카테고리의 ID

■ **getCategoryID**

Prototype	`public int getCategoryID(String fullPath) throws OZCPException`
Definition	전체 경로에 해당하는 카테고리의 ID를 가져옵니다.
Argument	*fullPath* 카테고리의 ID를 가져올 카테고리 경로

■ **updateUpperCategoryId**

Prototype	`public void updateUpperCategoryId(int cid, int upperCid) throws OZCPException`	
Definition	지정한 카테고리의 상위 카테고리 ID를 변경합니다.	
Argument	*cid*	카테고리 ID
	upperCid	상위 카테고리 ID

■ **updateCategoryName**

Prototype	`public void updateCategoryName(int cid, String categoryName) throws OZCPException`	
Definition	지정한 카테고리 ID에 해당하는 카테고리의 이름을 변경합니다.	
Argument	*cid*	이름을 변경할 카테고리의 ID
	categoryName	변경할 카테고리 이름

■ **getItemCountInCategory**

Prototype	`public int getItemCountInCategory(int cid) throws OZCPException`
Definition	지정한 카테고리에 속해 있는 모든 아이템의 개수를 가져옵니다.
Argument	*cid* 아이템의 개수를 얻을 카테고리의 ID

■ getCategoryListInCategory

Prototype	public OZRepositoryCategory[] getCategoryListInCategory(int cid) throws OZCPException public OZRepositoryCategory[] getCategoryListInCategory(int cid, int uid, byte perm) throws OZCPException	
Definition	지정한 카테고리에 포함된 모든 하위 카테고리의 정보를 가져옵니다.	
Argument	*cid*	카테고리 ID
	uid	사용자 ID
	perm	권한 정보

■ getCategoryInfo

Prototype	public OZRepositoryCategory getCategoryInfo(int cid) throws OZCPException	
Definition	지정한 카테고리의 정보를 가져옵니다.	
Argument	*cid*	정보를 가져올 카테고리 ID

■ getCategoryListAuthToUser

Prototype	public OZRepositoryCategory[] getCategoryListAuthToUser(int uid, int cid, byte perm) throws OZCPException	
Definition	지정한 사용자 ID에 해당하는 사용자에게 perm 이상의 권한이 부여된 모든 카테고리의 정보를 가져옵니다.	
Argument	*uid*	사용자 ID
	cid	카테고리 ID
	perm	권한 정보

■ getCategoryListAuthToGroup

Prototype	public OZRepositoryCategory[] getCategoryListAuthToGroup(int gid, int cid, byte perm) throws OZCPException	
Definition	지정한 그룹 ID에 해당하는 그룹에게 perm 이상의 권한이 부여된 모든 카테고리의 정보를 가져옵니다.	
Argument	*gid*	그룹 ID
	cid	카테고리 ID
	perm	권한 정보

/ / CheckInOut

■ **checkOutItem**

Prototype	public void checkOutItem(int itemid, int uid, String checkOutFolder) throws OZCPException	
Definition	지정한 아이템을 지정한 사용자 ID로 체크 아웃할 폴더에 체크 아웃을 합니다.	
Argument	*itemid*	체크 아웃할 아이템의 ID
	uid	체크 아웃할 사용자의 ID
	checkOutFolder	체크 아웃할 폴더 이름

■ **undoCheckOutItem**

Prototype	public void undoCheckOutItem(int itemId, int uid) throws OZCPException	
Definition	지정한 아이템을 지정한 사용자 ID로 체크 아웃 취소합니다.	
Argument	*itemId*	체크 아웃 취소할 아이템의 ID
	uid	체크 아웃 취소할 사용자의 ID

■ **checkInItem**

Prototype	public void checkInItem(boolean keepChkOut, int itemId, int uid, InputStream itemIn) throws OZCPException	
Definition	지정한 아이템을 지정한 사용자 ID로 체크인합니다.	
Argument	*keepChkOut*	체크 아웃 상태를 유지할지 여부
	itemId	체크인할 아이템의 ID
	uid	체크인할 사용자의 ID
	itemIn	아이템의 입력 스트림

■ **isCheckOutUser**

Prototype	public boolean isCheckOutUser(int itemid, int uid) throws OZCPException	
Definition	지정한 사용자가 지정한 아이템을 체크 아웃했는지 여부를 확인합니다.	
Argument	*itemId*	아이템 ID
	uid	체크 아웃한 사용자인지 체크할 사용자 ID

/ / History

■ **getSpecifiedVersionItem**

Prototype	public InputStream getSpecifiedVersionItem(int itemId, int itemVersion) throws OZCPException	
Definition	지정한 아이템 ID에 해당하는 아이템 중 지정한 버전의 아이템을 가져옵니다.	
Argument	*itemId*	가져올 아이템의 ID
	itemVersion	가져올 아이템의 버전

■ **getHistoryItemList**

Prototype	public OZRepositoryHistory[] getHistoryItemList(int itemId) throws OZCPException	
Definition	지정한 아이템의 히스토리 정보를 가져옵니다.	
Argument	*itemId*	히스토리 정보를 가져올 아이템의 ID

■ **clearHistoryItem**

Prototype	public void clearHistoryItem(int itemId, int itemVersion) throws OZCPException	
Definition	지정한 아이템에 대해 특정 버전의 히스토리를 삭제합니다.	
Argument	*itemId*	히스토리를 삭제할 아이템의 ID
	itemVersion	히스토리를 삭제할 버전

■ **rollBackItem**

Prototype	public void rollBackItem(int itemId, int itemVersion) throws OZCPException	
Definition	지정한 아이템을 지정한 버전으로 복원시킵니다.	
Argument	*itemId*	복원시킬 아이템의 ID
	itemVersion	복원시킬 버전

관련 Class

■ **OZRepositoryUser(oz.framework.repository.OZRepositoryUser)**

오즈 사용자에 대한 기본적인 정보 즉, 사용자 이름, 사용자 아이디, 패스워드, 속한 그룹 등을 가지고 있는 클래스입니다.

- 메소드
  - getUserName

Prototype	`public String getUserName()`
Definition	사용자 이름을 가져옵니다.

- getUserID

Prototype	`public int getUserID()`
Definition	사용자 ID를 가져옵니다.

- getGroupList

Prototype	`public java.util.Vector getGroupList()`
Definition	그룹 정보를 가져옵니다.

- getDescription

Prototype	`public String getDescription()`
Definition	사용자 설명 내용을 가져옵니다.

- getPassWord

Prototype	`public String getPassWord()`
Definition	사용자 패스워드를 가져옵니다.

- getPermission

Prototype	`public byte getPermission()`
Definition	사용자 권한 정보를 가져옵니다. • 0 : None(권한 없음) • 1 : View(목록 조회만 가능) • 3 : Read(아이템 읽기 가능) • 7 : Write(아이템 삭제 가능)

- getDirectPermission

Prototype	`public byte getDirectPermission()`
Definition	사용자 권한 정보를 가져옵니다.

- getInDirectPermission

Prototype	`public byte getInDirectPermission()`
Definition	사용자 권한 정보를 가져옵니다.

- getIsLoggedIn

Prototype	`public boolean getIsLoggedIn()`
Definition	사용자가 서버에 접속되어 있는지 여부를 확인합니다.

- getSessionID

Prototype	`public int getSessionID()`
Definition	사용자의 세션 ID를 가져옵니다.

- getIsLoginEnabled

Prototype	`public boolean getIsLoginEnabled()`
Definition	사용자가 접속 가능한 사용자인지 확인합니다.

■ **OZRepositoryUser[](oz.framework.repository.OZRepositoryUser[])**

오즈 사용자 리스트 정보를 가지고 있는 클래스입니다.

■ **OZRepositoryGroup(oz.framework.repository.OZRepositoryGroup)**

오즈 그룹에 대한 기본적인 정보 즉, 그룹명, 그룹 아이디, 상위 그룹 아이디, 관리자 아이디 등을 가지고 있는 클래스입니다.

- 메소드

- getGroupName

Prototype	`public String getGroupName()`
Definition	그룹 이름을 가져옵니다.

- getGroupID

Prototype	`public int getGroupID()`
Definition	그룹 ID를 가져옵니다.

- getParentGroupID

Prototype	`public int getParentGroupID()`
Definition	상위 그룹 ID를 가져옵니다.

- getGroupAdminList

Prototype	`public java.util.Vector getGroupAdminList()`

Definition	그룹 관리자 정보를 가져옵니다.

- getDirectPermission

Prototype	`public byte getDirectPermission()`
Definition	그룹 권한 정보를 가져옵니다.

- getInDirectPermission

Prototype	`public byte getInDirectPermission()`
Definition	그룹 권한 정보를 가져옵니다.

- getPermission

Prototype	`public byte getPermission()`
Definition	그룹 권한 정보를 가져옵니다.

- getDescription

Prototype	`public String getDescription()`
Definition	그룹 설명 내용을 가져옵니다.

- getFullPath

Prototype	`public String getFullPath()`
Definition	그룹 전체 경로를 가져옵니다.

■ **OZRepositoryGroup[](oz.framework.repository.OZRepositoryGroup[])**

오즈 그룹 리스트 정보를 가지고 있는 클래스입니다.

■ **OZRepositoryItem(oz.framework.repository.OZRepositoryItem)**

오즈 아이템에 대한 기본적인 정보 즉, 아이템명, 아이템 아이디, 아이템 타입, 카테고리명 등을 가지고 있는 클래스입니다.

- 메소드

- getItemName

Prototype	`public String getItemName()`
Definition	아이템 이름을 가져옵니다.

- getItemId

Prototype	`public int getItemId()`
Definition	아이템 ID를 가져옵니다.

- getItemType

Prototype	`public int getItemType()`
Definition	아이템 타입을 가져옵니다. • 10000 : ODI_FILE • 20001 : OZR_FILE • 20002 : SDM_FILE • 20003 : USDM_FILE • 20004 : OZD_FILE • 30001 : UIMG_FILE

- getDescription

Prototype	`public String getDescription()`
Definition	아이템 설명 내용을 가져옵니다.

- getCheckOutUserId

Prototype	`public int getCheckOutUserId()`
Definition	체크 아웃한 사용자 ID를 가져옵니다.

- getCheckOutUserName

Prototype	`public String getCheckOutUserName()`
Definition	체크 아웃한 사용자 이름을 가져옵니다.

- getCheckOutFolder

Prototype	`public String getCheckOutFolder()`
Definition	체크 아웃한 아이템의 폴더 이름을 가져옵니다.

- getUpdateTime

Prototype	`public java.util.Date getUpdateTime()`
Definition	아이템을 갱신한 시간 정보를 가져옵니다.

- getIsCheckedOut

Prototype	`public boolean getIsCheckedOut()`
Definition	체크 아웃 여부를 확인합니다.

- getDirectPermission

Prototype	`public byte getDirectPermission()`
Definition	권한 정보를 가져옵니다.

- getInDirectPermission

Prototype	`public byte getInDirectPermission()`
Definition	권한 정보를 가져옵니다.

- getAdminList

Prototype	`public java.util.Vector getAdminList()`
Definition	관리자 정보를 가져옵니다.

- getCategoryList

Prototype	`public java.util.Vector getCategoryList()`
Definition	카테고리 정보를 가져옵니다.

■ **OZRepositoryItem[](oz.framework.repository.OZRepositoryItem[])**

오즈 아이템 리스트 정보를 가지고 있는 클래스입니다.

■ **OZRepositoryCategory(oz.framework.repository.OZRepositoryCategory)**

오즈 리파지토리의 카테고리 대한 기본적인 정보를 가지고 있는 클래스입니다.

- 메소드

- getCategoryName

Prototype	`public String getCategoryName()`
Definition	카테고리 이름을 가져옵니다.

- getCategoryID

Prototype	`public int getCategoryID()`
Definition	카테고리 ID를 가져옵니다.

- getParentCategoryID

Prototype	`public int getParentCategoryID()`
Definition	상위 카테고리의 ID를 가져옵니다.

- getCategoryAdminList

Prototype	`public java.util.Vector getCategoryAdminList()`
Definition	카테고리의 관리자 정보를 가져옵니다.

- getDirectPermission

Prototype	`public byte getDirectPermission()`
Definition	권한 정보를 가져옵니다.

- getInDirectPermission

Prototype	`public byte getInDirectPermission()`
Definition	권한 정보를 가져옵니다.

- getPermission

Prototype	`public byte getPermission()`
Definition	권한 정보를 가져옵니다.

- getDescription

Prototype	`public String getDescription()`
Definition	카테고리의 설명 내용을 가져옵니다.

- getFullPath

Prototype	`public String getFullPath()`
Definition	카테고리의 전체 경로를 가져옵니다.

■ **OZRepositoryCategory[](oz.framework.repository.OZRepositoryCategory[])**

오즈 카테고리 리스트 정보를 가지고 있는 클래스입니다.

■ **OZRepositoryHistory(oz.framework.repository.OZRepositoryHistory)**

오즈 아이템 히스토리의 정보를 가지고 있는 클래스입니다.

- 메소드

▪ getHistoryItemPath

Prototype	`public String getHistoryItemPath()`
Definition	아이템의 경로를 가져옵니다.

▪ getHistoryItemVersion

Prototype	`public int getHistoryItemVersion()`
Definition	아이템의 버전 정보를 가져옵니다.

▪ getHistoryDate

Prototype	`public String getHistoryDate()`
Definition	아이템의 히스토리 날짜 정보를 가져옵니다.

▪ getHistoryCheckInUser

Prototype	`public String getHistoryCheckInUser()`
Definition	아이템의 체크인 사용자 정보를 가져옵니다.

■ **OZRepositoryHistory[](oz.framework.repository.OZRepositoryHistory[])**

오즈 아이템 히스토리 정보를 가지고 있는 클래스입니다.

Sample : RepositorySample.java

```
package sample;

import oz.framework.api.Repository;
import oz.util.SortProperties;
import oz.framework.repository.OZRepositoryItem;
import oz.framework.repository.OZRepositoryUser;
import oz.framework.repository.OZRepositoryHistory;
import oz.framework.repository.OZRepositoryGroup;
import oz.framework.repository.OZRepositoryCategory;

import oz.framework.cp.OZCPException;
import oz.dm.hc.HCDataModule;
```

```
import oz.dm.DMConst;
import java.io.*;
import java.net.*;
import org.apache.log4j.*;
import java.io.IOException;
import java.util.*;

public class RepositorySample {
   private static Repository repository = null;

   public static void main(String[] args) {
      //아래 라인을 주석 처리하면 사용자가 지정하지 않은 로그가 나타나지 않습니다.
      BasicConfigurator.configure();

      // OZServer Info.
      //*
      // Daemon
      String IP = "127.0.0.1"; //서버가 설치되어 있는 호스트 컴퓨터의 IP
      int PORT = 8003; //서버가 사용하는 TCP 포트
      /*/
      // Servlet
      String URL  = "http://www.oz.com/oz/server"; //Servlet 에 접근가능한 URL
      /**/
      // User Info.
      String ID = "admin"; //default
      String PWD = "admin"; //default


      try {
         // Daemon
         repository = new Repository(IP, PORT, ID, PWD, false, false);
         /*/
         // Servlet
         repository = new Repository(URL, ID, PWD, false, false);
         /**/

         repositoryConfiguration();
         userTest();
         groupTest();
         itemTest();
      }
      catch (Exception e)
      {
         e.printStackTrace();
      }
   }
   private static void repositoryConfiguration() throws Exception {
      //Repository configuration 정보 가져오기.
```

```
    System.out.println("Repository.getRepositoryConfig()");
    SortProperties props = repository.getRepositoryConfig();
    props.list(System.out);

    //Repository Configuration 설정
    SortProperties prop = new SortProperties();
    prop.setProperty("REPOSITORY_TYPE", "RDB"); //RDB, FILESYSTEM, USER
    prop.setProperty("REPOSITORY_FILE_PATH", "c:/temp_repository");
    prop.setProperty("REPOSITORY_ITEM_NUMBER_PER_DIRECTORY", "100");
    prop.setProperty("REPOSITORY_HISTORY_ITEM_VALID_DAYS", "20");
    //repository.setRepositoryConfig(prop);
}

private static void historyTest(int itemid) throws Exception {
    final int itemVersion = 0;

    // itemVersion, itemid 에 해당하는 아이템 가져오기
    System.out.println("Repository.getSpecifiedVersionItem(
       int itemid, int itemVersion)");
    InputStream in = repository.getSpecifiedVersionItem(
       itemid, itemVersion);
    download(repository.getSpecifiedVersionItem(itemid, itemVersion),
       "getSpecifiedVersionItem" +
       repository.getItemInfo(itemid).getItemName());

    // itemid 아이템의 히스토리 정보 가져오기
    System.out.println("Repository.getHistoryItemList(int itemid)");
    OZRepositoryHistory[] historyInfoList =
       repository.getHistoryItemList(itemid);
    if (historyInfoList != null) {
       for (int i = 0; i < historyInfoList.length; i++) {
          OZRepositoryHistory historyInfo = historyInfoList[i];
          System.out.println("HistoryDate : " +
                         historyInfo.getHistoryDate());
          System.out.println("HistoryItemPath : " +
                         historyInfo.getHistoryItemPath());
          System.out.println("HistoryItemVersion : " +
                         historyInfo.getHistoryItemVersion());
       }
    }

   // 해당 Item 의 지정된 version 의 history 를 삭제합니다.
   // history 를 삭제할 경우 아래의 복원부분 실행 불가능
   // System.out.println("Repository.clearHistoryItem(
   //    int itemid, int itemVersion)");
   // repository.clearHistoryItem(itemid, itemVersion);

   // 변경된 Item 을 지정된 itemVersion 으로 복원합니다.
```

```
   System.out.println("Repository.rollBackItem(
      int itemid, int itemVersion)");
   repository.rollBackItem(itemid, itemVersion);
}

private static void checkInOutTest(int itemid) throws Exception{
    final int uid = repository.getUserIdbyName("admin");
    String fileName = "check in out test.txt";
    createFile(fileName);

    // User(uid)가 Item(itemid)를 checkout 합니다.
    System.out.println("Repository.checkOutItem(
        int itemid, int uid, String checkoutFolder)");
    String checkoutFolder = "..";
    repository.checkOutItem(itemid, uid, checkoutFolder);
    // uid 가 itemid 에 대해 checkout 을 했으면, checkout 을 cancel 합니다.
    // System.out.println("Repository.undoCheckOutItem(
    //   int itemid, int uid)");
    // repository.undoCheckOutItem(itemid, uid);

    // item checkin 합니다. keepChkOut 은 check-in 후 check-out 상태를 유지 여부
    System.out.println(
       "Repository.checkInItem(boolean keepChkOut, int itemid, int uid,
            InputStream item_in)");
    boolean keepChkOut = false;
    FileInputStream in = new FileInputStream(fileName);
    repository.checkInItem(keepChkOut, itemid, uid, in);
    in.close();

    // uid user 가 itemid 를 check out 했는지 체크합니다.
    System.out.println("Repository.isCheckOutUser(int itemid, int uid)");
    System.out.println("Does UID[" + uid + "] check out the item[" +
       itemid + "] ? " + repository.isCheckOutUser(itemid, uid));
    removeFile(fileName);
}

private static void categoryTest() throws Exception {
    final int uid = repository.getUserIdbyName("admin");
    int gid = repository.getGroupIdOfUser(uid);
    int cid, parent_cid;
    final byte authREAD = 0x02, authVIEW = 0x01;
    String categoryName = null;
    OZRepositoryCategory[] categoryInfoList = null;

    // 카테고리를 생성합니다.
    System.out.println("Repository.createCategory(String categoryPath)");
    categoryName = "/Poultry";
    cid = repository.createCategory(categoryName);
```

```
System.out.println(
   "categoryPath [categoryID] : " + categoryName + "[" + cid + "]");

// 상위 카테고리 upper_cid 하에 카테고리를 category_name 으로 생성합니다.
System.out.println("Repository.createCategory(String category_name,
    int upper_cid)");
String childCategoryName = "Chickens";
parent_cid = cid;
cid = repository.createCategory(childCategoryName, parent_cid);
System.out.println("New new_category_name [new_categoryID] : " +
               childCategoryName + "[" + cid + "]");

// 카테고리 cid 를 삭제합니다.
System.out.println("Repository.deleteCategory(int cid)");
repository.deleteCategory(cid);
cid = parent_cid;

// Full Path 에 존재하는 카테고리 아이디 값을 가져옵니다.
System.out.println("Repository.getCategoryID(String fullPath)");
System.out.println("the full Path [" + categoryName +
    "]'s category ID : " + repository.getCategoryID(categoryName));

// 카테고리 이름을 변경합니다.
System.out.println("Repository.updateCategoryName(int cid,
    String categoryName)");
categoryName = "Fishes";
System.out.println("Before Update CategoryName : " +
       (repository.getCategoryInfo(cid)).getCategoryName());
repository.updateCategoryName(cid, categoryName);
System.out.println("After Update CategoryName : " +
       (repository.getCategoryInfo(cid)).getCategoryName());
// 상위 카테고리를 변경합니다.
System.out.println("Repository.updateUpperCategoryId(
       int cid, int uppercid)");
int new_cid = repository.createCategory("/Category Test");
System.out.println("Before Update UpperCategoryID : " +
        repository.getCategoryInfo(cid)).getParentCategoryID());
repository.updateUpperCategoryId(cid, new_cid);
System.out.println("After Update UpperCategoryID : " +
        (repository.getCategoryInfo(cid)).getParentCategoryID());

// cid 카테고리에 속한 아이템의 개수를 가져옵니다.
System.out.println("Repository.getItemCountInCategory(String cid)");
System.out.println("Item count in the category id [" + cid + "] : " +
               repository.getItemCountInCategory(cid) + "");

// cid 카테고리에 속한 하위 카테고리 리스트 정보를 가져옵니다.
System.out.println("Repository.getCategoryListInCategory(int cid) ");
```

```
    categoryInfoList = repository.getCategoryListInCategory(0);
    showCategoryInfoList(categoryInfoList);

    // 카테고리에서 사용자 id 가 보기 이상의 권한을 가지고 있는
    // 하위 카테고리 리스트 정보를 가져옵니다.
    System.out.println("Repository.getCategoryListInCategory(
           int categoryId, int userId, byte perm)");
    categoryInfoList = repository.getCategoryListInCategory(cid, uid,
            authVIEW);
    showCategoryInfoList(categoryInfoList);

    // 카테고리에서 그룹 gid 에게 보기 이상의 권한을 가지고 있는
    // 하위 카테고리 리스트 정보를 가져옵니다.
    System.out.println(
       "Repository.getCategoryListInCategoryAuthGroup(int categoryId,
            int groupId, byte perm)");
    categoryInfoList =
       repository.getCategoryListInCategoryAuthGroup(cid, gid, authVIEW);
    showCategoryInfoList(categoryInfoList);

    // 카테고리 id 에 해당하는 정보를 가져옵니다.
    System.out.println("Repository.getCategoryInfo(int cid))");
    showCategory(repository.getCategoryInfo(cid));

    // 카테고리 내에서 사용자에게 읽기 이상의 권한이 부여된
    // 하위 카테고리 리스트 정보를 가져옵니다.
    System.out.println(
       "Repository.getCategoryListAuthToUser(int userId,int categoryId,
            byte perm) ");
    categoryInfoList = repository.getCategoryListAuthToUser(uid, cid,
            authREAD);
    showCategoryInfoList(categoryInfoList);

    // 카테고리내에서 그룹에게 읽기 이상의 권한이 부여된
    // 하위 카테고리 리스트 정보를 가져옵니다
    System.out.println("Repository.getCategoryListAuthToGroup(int groupId,
           int categoryId, byte perm) ");
    categoryInfoList = repository.getCategoryListAuthToGroup(gid, cid,
           authREAD);
    showCategoryInfoList(categoryInfoList);

    repository.deleteCategory(cid);
    repository.deleteCategory(new_cid);
}

private static void showCategoryInfoList(
    OZRepositoryCategory[] categoryList) {
    if(categoryList == null)
```

```
            return;
         for(int i=0; i < categoryList.length; i++) {
            showCategory(categoryList[i]);
         }
      }

      private static void showCategory(OZRepositoryCategory c) {
         System.out.println("CategoryId : " + c.getCategoryID());
         System.out.println("CategoryName : " + c.getCategoryName());
         System.out.println("CategoryUpperId : " + c.getParentCategoryID());
         System.out.println("Permission : " + c.getPermission());
      }

      private static void itemListTest(int itemid) throws Exception {
         FileOutputStream fos = null;
         InputStream in = null;
         OZRepositoryItem[] itemInfoList = null;
         OZRepositoryUser[] userInfoList = null;
         OZRepositoryCategory[] categoryInfoList = null;
         OZRepositoryGroup[] groupList = null;
         final String userName = "admin";
         String itemName = repository.getItemInfo(itemid).getItemName();
         int uid = repository.getUserIdbyName(userName);
         int cid = repository.getCategoryIdOfItem(itemid);
         int gid = repository.getGroupIdOfItem(itemid);
         String categoryName = "/" +
repository.getCategoryInfo(cid).getCategoryName();
         final byte authRW = 0x07, authREAD = 0x02;
         byte[] buf = null;

/*************************************************************************
         지원되지 않음.
         // 아이템 리스트 정보를 가져옵니다.
         System.out.println("Repository.getItemList()");
         itemInfoList = repository.getItemList();
         showItemInfoList(itemInfoList);
*************************************************************************/

         // 아이템 정보를 가져옵니다.
         System.out.println("Repository.getItemInfo(int itemid)");
         OZRepositoryItem itemInfo = repository.getItemInfo(itemid);
         showItem(itemInfo);

         // 카테고리 id 에 속한 아이템 리스트 정보를 가져옵니다.
         System.out.println("Repository.getItemListInCategory(int cid)");
         itemInfoList = repository.getItemListInCategory(cid);
         showItemInfoList(itemInfoList);
```

```
// 카테고리에 속한 아이템 리스트 정보를 가져옵니다.
System.out.println(
   "Repository.getItemListInCategory(String categoryName)");
itemInfoList = repository.getItemListInCategory(categoryName);
showItemInfoList(itemInfoList);

// 카테고리명 에 속한 아이템등 중에서 사용자 userId 에게 읽기쓰기 이상의
// 권한이 주어진 아이템 리스트 정보를 가져옵니다.
System.out.println(
   "Repository.getItemListInCategory(String cName,int userid,
    byte perm)");
itemInfoList = repository.getItemListInCategory(categoryName, uid,
    authRW);
showItemInfoList(itemInfoList);

// 카테고리 categoryId 에 속한 아이템등 중에서 사용자 userId 에게 읽기 이상의
// 권한이 주어진 아이템 리스트 정보를 가져옵니다.
System.out.println("Repository.getItemListInCategory(int cid, int
     userid, byte perm)");
itemInfoList = repository.getItemListInCategory(cid, uid, authREAD);
showItemInfoList(itemInfoList);

// 카테고리 ID 에 속한 아이템등 중에서 그룹 GroupID 에게 읽기 이상의
// 권한이 주어진 아이템 리스트 정보를 가져옵니다.
System.out.println("Repository.getItemListInCategoryAuthGroup(int cid,
    int groupid, byte perm)");
itemInfoList = repository.getItemListInCategoryAuthGroup(cid, gid,
     authREAD);
showItemInfoList(itemInfoList);

// 카테고리에 속한 아이템등 중에서 그룹 GroupID 에게 읽기 이상의
// 권한이 주어진 아이템 리스트 정보를 가져옵니다.
System.out.println("Repository.getItemListInCategoryAuthGroup” +
   ”(String categoryName, int groupid, byte perm)");
itemInfoList = repository.getItemListInCategoryAuthGroup(categoryName,
    gid, authREAD);
showItemInfoList(itemInfoList);

// 사용자 uid 에게 읽기 이상의 권한이 부여된 아이템 리스트 정보를 가져옵니다.
System.out.println("Repository.getItemListAuthToUser(int uid,
    byte perm)");
itemInfoList = repository.getItemListAuthToUser(uid, authREAD);
showItemInfoList(itemInfoList);

// 그룹 gid 에게 읽기 이상의 권한이 부여된 아이템 리스트 정보를 가져옵니다.
System.out.println("Repository.getItemListAuthToGroup(int gid,
    byte perm)");
itemInfoList = repository.getItemListAuthToGroup(gid, authREAD);
```

```
      showItemInfoList(itemInfoList);

/*************************************************************************
       지원되지 않음
      // 아이템 인덱스로 아이템 정보를 가져옵니다.
      System.out.println(
         "Repository.findItemByItemIndex(String[] itemIndex, int[] oper)");
      String[] itemIndex = {"index test 1", "index test 2", "index test 3"};
      int[] oper = {2, 2}; //1 => AND, 2 => OR
      itemInfoList = repository.findItemByItemIndex(itemIndex, oper);
      showItemInfoList(itemInfoList);
*************************************************************************/
   }

   private static void showItemInfoList(OZRepositoryItem[] itemInfoList) {
      if(itemInfoList == null)
         return;
      System.out.println(
         "[i]CategoryName:ItemName:ItemID-------------------------------");
      OZRepositoryItem item;
      for(int i=0; i < itemInfoList.length; i++) {
         item = itemInfoList[i];
         Vector v = item.getCategoryList();
         Iterator iter = v.iterator();
         while(iter.hasNext()) {
            System.out.println("[" + i + "]" + iter.next() + ":" +
                           item.getItemName() + ":" + item.getItemId());
         }
      }
   }

   private static void showItem(OZRepositoryItem item) {
      Vector v = item.getCategoryList();
         Iterator iter = v.iterator();
         while(iter.hasNext()) {
            System.out.println(" " + iter.next() + ":" +
                           item.getItemName() + ":" +
                           item.getItemId());
         }
   }

   private static void infoByItemTest(int itemid) throws Exception {
      // 해당 아이템의 카테고리 구하기
      System.out.println("Repository.getCategoryIdOfItem(int itemid)");
      System.out.println("the ItemID[" + itemid + "]'s category ID : " +
                     repository.getCategoryIdOfItem(itemid));
      System.out.println(
         "Repository.updateCategoryIdOfItem(int cid, int itemid)");
```

```
    int cid = repository.getCategoryIdOfItem(itemid);
    System.out.println("Before Update CategoryID : " +
                    repository.getCategoryIdOfItem(itemid));
    repository.updateCategoryIdOfItem(cid, itemid);
    System.out.println("After Update CategoryID : " +
                    repository.getCategoryIdOfItem(itemid));
}

private static void itemTest() throws Exception {
    final String itemFileName = "대용량 100 만.odi";
    String itemName = itemFileName;
    String categoryName = "/Item Test";
    String itemDescription = "Test of item creation";
    final int itemType = 10000; // OZItemInfo.ODI_FILE

    int cid = repository.createCategory(categoryName);
    int itemid;

    // 신규 아이템을 생성하여 그 아이템 아이디를 return 합니다.
    System.out.println(
       "Repository.createItem(name, type, desc, cid, input_stream)");
    itemid = repository.createItem(itemFileName, itemType,
                            itemDescription, cid,
                            new FileInputStream(itemFileName));
    System.out.println("item_name [iid] : " + itemName + "[" +
                       itemid + "]");

    // 신규 아이템을 카테고리이름을 기준으로 생성하여 아이디를 return
    // System.out.println(
    // "Repository.createItem(name, type, desc, category_name,
    //                         input_stream)");
    // itemid = repository.createItem(itemFileName, itemType,
    //                               itemDescription, categoryName,
    //                               new FileInputStream(itemFileName));
    // System.out.println("item_name [iid] : " + itemFileName + "[" +
    //                    itemid + "]");

    // 아이템명, 아이템 타입, 아이템이 속한 카테고리 명에 존재하는
    // 그 아이템의 ID 를 가져옵니다
    System.out.println("Repository.getItemId(String item_name,
                       int item_type, Sring categoryName)");
    itemid = repository.getItemId(itemName, itemType, categoryName);
    System.out.println("Item ID [" + itemid + "]");

    // 아이템명, 아이템 타입, 아이템이 속한 카테고리 id 에 존재하는
    // 그 아이템의 ID 를 가져옵니다
    System.out.println(
       "Repository.getItemId(String item_name, int item_type, int cid)");
```

```
      itemid = repository.getItemId(itemName, itemType, cid);
      System.out.println("Item ID [" + itemid + "]");

      // 아이템 파일을 가져옵니다.
      System.out.println("Repository.getItem(int itemid, int cid)");
      System.out.println("Read the item from server and create a new file.");
      download(repository.getItem(itemid, cid),"getItem" + itemName);

      // 아이템 이름을 변경합니다.
      System.out.println("Repository.updateItemName(int itemId,
                        String itemName) ");
      itemName = "Changed item name.odi";
      System.out.println("Before Update ItemName : " +
                     (repository.getItemInfo(itemid)).getItemName());
      repository.updateItemName(itemid, itemName);
      System.out.println("After Update ItemName : " +
                     (repository.getItemInfo(itemid)).getItemName());

      // 아이템 파일을 가져옵니다.
      System.out.println("Repository.getDirectItem” +
         ”(String itemName, int itemType, String categoryName,
           boolean compressedItem)");
      System.out.println("Read the item from server and create a new file.");
      boolean compress = true;
      download(repository.getDirectItem(itemName, itemType, categoryName,
             compress), "getDirectItem_compressed" + itemName);

      // 아이템 파일을 가져옵니다.
      System.out.println("Repository.getDirectItem(String itemName,
                         int itemType, String categoryName)");
      System.out.println("Read the item from server and create a new file.");
      download(repository.getDirectItem(itemName, itemType, categoryName),
             "getDirectItem_uncompressed" + itemName);
      FileInputStream fis;

/*************************************************************************
       현재 지원되지 않음
      // 기존 아이템을 갱신.
      System.out.println("Repository.updateItem(int itemId,
                        InputStream input)");
      fis = new FileInputStream(itemFileName);
      repository.updateItem(itemid, fis);
      fis.close();
      // 기존 아이템을 갱신.
      System.out.println(
         "Repository.updateDirectItem(String itemName, int itemType,
                              String categoryName, InputStream input)");
      fis = new FileInputStream(itemFileName);
```

```
      repository.updateDirectItem(itemName, itemType, categoryName, fis);
      fis.close();
*************************************************************************/

      // 아이템이 카테고리 내에 있는지 검사
      System.out.println("Repository.hasItemInRepository(
                  String itemName, int itemType, String categoryName)");
      System.out.println("Does item[" + itemName + "] exist in category[" +
                        categoryName + "] ? " +
                        repository.hasItemInRepository(itemName, itemType,
                        categoryName));

      infoByItemTest(itemid);
      itemListTest(itemid);
      categoryTest();
      checkInOutTest(itemid);
      historyTest(itemid);

      // 아이템을 삭제합니다.
      System.out.println("Repository.deleteItem(int itemid)");
      repository.deleteItem(itemid);

      repository.deleteCategory(cid);
   }

   private static void download(InputStream in, String fileName)
   throws Exception {
      byte[] buf = new byte[in.available()];
      in.read(buf);
      in.close();
      FileOutputStream fos = new FileOutputStream(fileName);
      fos.write(buf);
      fos.flush();
      fos.close();
   }

   private static void groupListTest(int gid) throws Exception {
      // 그룹 gid 의 하위 그룹 리스트 정보를 가져옵니다.
      System.out.println("Repository.getGroupListInGroup(int gid))");
      OZRepositoryGroup[] groupInfoList = repository.getGroupListInGroup(0);
      showGroupList(groupInfoList);

      // 그룹 gid 의 정보를 가져옵니다.
      System.out.println("Repository.getGroupInfo(int gid))");
      OZRepositoryGroup groupInfo = repository.getGroupInfo(gid);
      showGroup(groupInfo);

      // 아이템 itemid 에 퍼미션 perm 이상의 권한을 가진 그룹정보를 가져옵니다.
```

```
    System.out.println("Repository.getGroupListAuthToItem(int itemid,
                        byte perm)");
    String sdmName = "SDM for test.sdm";
    final int itemType = 20002; //OZFrameworkConst.SDM_FILE;
    createFile(sdmName);
    int cid = repository.createCategory("/User List Test");
    int itemid = repository.createItem(sdmName, itemType, "", cid,
                                       new FileInputStream(sdmName));
    final byte authREAD = 0x02;
    groupInfoList = repository.getGroupListAuthToItem(itemid, authREAD);
    showGroupList(groupInfoList);

    // 카테고리 categoryId 에 읽기 이상의 권한을 가진 그룹정보를 가져옵니다.
    System.out.println("Repository.getGroupListAuthToCategory(
                        int categoryId, byte perm) ");
    groupInfoList = repository.getGroupListAuthToCategory(cid, authREAD);
    showGroupList(groupInfoList);

    repository.deleteItem(itemid);
    repository.deleteCategory(cid);
    removeFile(sdmName);
}

private static void showGroupList(OZRepositoryGroup[] groupList) {
    if(groupList == null)
        return;
    for(int i=0; i < groupList.length; i++) {
        showGroup(groupList[i]);
    }
}

private static void showGroup(OZRepositoryGroup group) {
    System.out.println("GroupId : " + group.getGroupID());
    System.out.println("GroupName : " + group.getGroupName());
    System.out.println("GroupUpperId : " + group.getParentGroupID());
    Vector v = group.getGroupAdminList();
    showGroupAdminUserName(v);
    System.out.println("Permission : " + group.getPermission());
}

private static void showGroupAdminUserName(Vector v) {
    Iterator it = v.iterator();
    while (it.hasNext()) {
        System.out.println("GroupAdminUserName : " + it.next());
    }

}
```

```
private static void groupAdminTest(int gid) throws Exception{
    // Group Admin 을 uid 로 변경합니다.
    System.out.println("Repository.updateUserGroupAdmin(int uid,
                        int gid)");
    int uid = repository.createUser("test id", "1234567", gid, "");
    System.out.println("Before : ");

      showGroupAdminUserName(repository.getGroupInfo(gid).getGroupAdminList
      ());

    repository.updateUserGroupAdmin(uid, gid);
    System.out.println("After : ");

    showGroupAdminUserName(repository.getGroupInfo(gid).getGroupAdminList()
    );

    // uid 가 gid 의 group admin 인지 체크합니다.
    System.out.println("Repository.isUserGroupAdmin(int uid, int gid)");
    System.out.println("Is this user[" + uid +
                    "] a group admin in Group[" + gid +
                    "] ? " + repository.isUserGroupAdmin(uid, gid));
    repository.deleteUser(uid);
}

private static void groupTest() throws Exception {
    // upper_gid 를 상위그룹으로 가지는 그룹 생성
    System.out.println("Repository.createGroup(String group_name,
                        int upper_gid)");
    String groupName = "forcs";
    int root_gid = 0;
    int parent_gid = root_gid;
    int gid = repository.createGroup(groupName, parent_gid);
    System.out.println("group_name [upper_gid] : " + groupName +
                        "[" + gid + "]");

    // 아이디가 gid 인 그룹을 삭제
    System.out.println("Repository.deleteGroup(int gid)");
    int temp_gid = repository.createGroup("Temporary group", root_gid);
    repository.deleteGroup(temp_gid);

    // 상위 그룹을 변경합니다.
    System.out.println("Repository.updateUpperGroupId(int gid,
                        int uppergid) ");
    parent_gid = repository.createGroup("Temporary group", root_gid);
    System.out.println("Before Update UpperGroupID : " +
                    (repository.getGroupInfo(gid)).getParentGroupID());
    repository.updateUpperGroupId(gid, parent_gid);
    System.out.println("After Update UpperGroupID : " +
```

```
                    (repository.getGroupInfo(gid)).getParentGroupID());

    // 그룹 이름을 변경합니다.
    System.out.println("Repository.updateGroupName(int groupId,
                        String groupName)");
    String newGroupName = "(주} 포시에스";
    System.out.println("Before Update GroupIDofUser : " +
                    (repository.getGroupInfo(gid)).getGroupName());
    repository.updateGroupName(gid, newGroupName);
    System.out.println("After Update GroupIDofUser : " +
                    (repository.getGroupInfo(gid)).getGroupName());

    groupAdminTest(gid);
    groupListTest(gid);

    repository.deleteGroup(parent_gid);
}

private static void createFile(String fileName) throws Exception {
    FileOutputStream fos = new FileOutputStream(new File(fileName));
    fos.write("dummy data".getBytes());
    fos.flush();
    fos.close();
}

private static void removeFile(String fileName) {
    try {
       File f = new File(fileName);
       f.delete();
    }
    catch(Exception e)
    {
    }
}

private static void userListTest(int uid) throws Exception {
    OZRepositoryUser userInfo = null;
    OZRepositoryUser[] userInfoList = null;
    int gid = repository.getGroupIdOfUser(uid);
    final String sdmName = "SDM for test.sdm";
    final int itemType = 20002; //OZRepositoryItem.SDM_FILE;
    createFile(sdmName);
    int cid = repository.createCategory("/User List Test");
    int itemid = repository.createItem(sdmName, itemType, "", cid,
                                        new FileInputStream(sdmName));
    final byte authREAD = 0x02;

    // 사용자 정보 리스트를 가져옵니다.
```

```
        System.out.println("Repository.getUserList()");
        userInfoList = repository.getUserList();
        showUserInfoList(userInfoList);

        // 그룹에 속하는 사용자 정보 리스트를 가져옵니다.
        System.out.println("Repository.getUserListInGroup(int gid)");
        userInfoList = repository.getUserListInGroup(gid);
        showUserInfoList(userInfoList);

        // 아이템에 읽기 이상의 권한을 가지고 있는 사용자 정보 리스트를 가져옵니다.
        System.out.println(
            "Repository.getUserListAuthToItem(int itemid, byte perm)");
        userInfoList = repository.getUserListAuthToItem(itemid, authREAD);
        showUserInfoList(userInfoList);

        // 카테고리에 읽기 이상의 권한을 가진 사용자 정보 리스트를 가져옵니다.
        System.out.println(
            "Repository.getUserListAuthToCategory(int categoryid, byte perm)");
        userInfoList = repository.getUserListAuthToCategory(cid, authREAD);
        showUserInfoList(userInfoList);

        repository.deleteItem(itemid);
        repository.deleteCategory(cid);
        removeFile(sdmName);
    }

    private static void showUserInfoList(OZRepositoryUser[] userInfoList) {
        if (userInfoList == null)
            return;
        OZRepositoryUser user;
        Iterator iter;
        System.out.println(
            "[i]GroupName:UserName:UserId----------------------------------");
        for (int i = 0; i < userInfoList.length; i++) {
            user = userInfoList[i];
            iter = user.getGroupList().iterator();
            while (iter.hasNext()) {
                System.out.println("[" + i + "]" + iter.next() + ":"
                                + user.getUserName() + ":" +
                                user.getUserID());
            }
        }
    }

    private static void userIDTest(int uid) throws Exception {
        // 사용자가 속한 Group ID 를 변경합니다.
        System.out.println(
            "Repository.updateGroupIdOfUser(int gid, int uid)");
```

```
    int old_gid = repository.getGroupIdOfUser(uid);
    int new_gid = repository.createGroup("group_for_test", old_gid);
    System.out.println("Before Update GroupIDofUser : " +
                    repository.getGroupIdOfUser(uid));
    repository.updateGroupIdOfUser(new_gid, uid);
    System.out.println("After Update GroupIDofUser : " +
                    repository.getGroupIdOfUser(uid));

    // 사용자명에 해당하는 사용자 아이디 얻기.
    System.out.println("Repository.getUserIdbyName(String user_name)");
    String userName = repository.getUserNameByID(uid);
    uid = repository.getUserIdbyName(userName);
    System.out.println("uid[" + uid + "]" + userName);
    repository.updateGroupIdOfUser(old_gid, uid);
    repository.deleteGroup(new_gid);
}

private static void userDescTest(int uid) throws Exception {
    // 사용자 설명을 변경합니다.
    System.out.println(
       "Repository.updateUserDescription(int uid, String description) ");
    String description = "안녕하세요. 포시에스입니다.";
    System.out.println("Before Update Description : " +
                    repository.getUserDescription(uid));
    repository.updateUserDescription(uid, description);
    System.out.println("After Update Description : " +
                    repository.getUserDescription(uid));
}

private static void userPwdTest(int uid) throws Exception {
    // 사용자 uid 의 패스워드가 pwd 가 맞는지 check 합니다.
    System.out.println("Repository.checkUserPwd(int uid, String pwd)");
    String pwd = "admin";

    // 사용자 uid 의 패스워드를 pwd 로 변경합니다.
    System.out.println("Repository.updateUserPwd(int uid, String pwd)");
    System.out.println("Before : UserName : "
                    + repository.checkUserPwd(uid, pwd));
    repository.updateUserPwd(uid, pwd);
    System.out.println("After : UserName : "
                    + repository.checkUserPwd(uid, pwd));
}

private static void userLoginTest(int uid) throws Exception {
    // default login user 를 loginVal 으로 변경합니다.
    System.out.println("Repository.updateLoginDefault(int loginDVal)");
    int loginDVal = uid; // uid 인 user 가 default login 으로 setting 됩니다.
    repository.updateLoginDefault(loginDVal);
```

```
    // userName 의 login 을 disable 합니다.
    System.out.println("Repository.disableUserLogin(String userName)");
    String userName = repository.getUserNameByID(uid);
    repository.disableUserLogin("'" + userName + "'");

    // userName 의 login 을 enable 합니다.
    System.out.println("Repository.enableUserLogin(String userName)");
    repository.enableUserLogin("'" + userName + "'");

    // 서버에 접속 되었는지 검사합니다.
    System.out.println("Repository.loginToServer()");
    System.out.println("loginToServer ? " + repository.loginToServer());

    // uid 에 해당되는 사용자를 logout 합니다.
    System.out.println("Repository.userLogout(int uid)");
    System.out.println("Before: Is this user[" + uid +
                    "] logged on now ? " + repository.isUserLogin(uid));
    //repository.userLogout(uid);
    System.out.println("After: Is this user[" + uid +
                    "] logged on now ? " + repository.isUserLogin(uid));

}

private static void userTest() throws Exception {

    // 새 사용자를 생성합니다.
    System.out.println(
       "Repository.createUser(String user_name, String pwd, int gid,
                              String description)");
    String userName = "forcs";
    String pwd = "111111";
    int gid = 0; // 루트 카테고리
    String description = "(주) 포시에스";
    int uid = repository.createUser(userName, pwd, gid, description);
    System.out.println("user_name [uid] : " + userName + "[" + uid + "]");

    // 사용자 이름을 변경합니다.
    System.out.println(
       "Repository.updateUserName(int userId, String userName)");
    String newUserName = "포시에스";
    System.out.println("Before : UserName : "
                    + repository.getUserNameByID(uid));
    repository.updateUserName(uid, newUserName);
    System.out.println("After : UserName : " +
                    repository.getUserNameByID(uid));

    userLoginTest(uid);
```

```
        userDescTest(uid);
        userIDTest(uid);
        userListTest(uid);
        userPwdTest(uid);

        // 사용자를 삭제합니다.
        System.out.println("Repository.deleteUser(int uid)");
        repository.deleteUser(uid);
    }
}
```

Class Service

Constructor Summary

- **Service(String ip, int port, String id, String pw, boolean bAutoLogin, boolean useUSL)**
- **Service(String url, String id, String pw, boolean bAutoLogin, boolean useUSL)**

Method Summary

- **void garbageCollect()**
- **void stop(boolean check)**
- **void restart()**

Constructor Detail

Prototype	**//Daemon 타입 - 오즈 서버 타입이 TCP Server인 경우** `public Service(String ip, int port, String id, String pw, boolean bAutoLogin, boolean useUSL)` **//Servlet 타입 - 오즈 서버 타입이 HTTP Server인 경우** `public Service(String url, String id, String pw, boolean bAutoLogin, boolean useUSL)`	
Argument	*url*	Servlet 타입 오즈 서버의 URL ex) String url = "http://127.0.0.1/oz/server";
	ip	Daemon 타입 오즈 서버의 IP ex) String ip = "127.0.0.1";
	port	Daemon 타입 오즈 서버의 포트 번호 ex) int port = 8003;

	id	사용자 아이디 ex) String id = "admin";
	pw	사용자 패스워드 ex) String pw = "admin";
	bAutoLogin	자동 로그인 여부 ex) boolean bAutoLogin = true;
	useUSL	USL 사용 여부 ex) boolean useUSL = false;

Method Detail

■ garbageCollect

Prototype	`public void garbageCollect() throws OZCPException`
Definition	별도의 메모리 수집 프로그램을 실행하여 JVM 상에서 사용되지 않는 메모리를 수거해 재사용 가능하게 해주는 garbage collection을 실행합니다.

■ stop

Prototype	`public void stop(boolean check) throws OZCPException`	
Definition	동작 중인 서버를 정상 종료시킵니다.	
Argument	*check*	강제 종료 여부 • true : 서버를 바로 종료 • false : 동작중인 작업이 완료된 후 서버를 종료

■ restart

Prototype	`public void restart() throws OZCPException`
Definition	동작 중인 서버를 정상 종료시키고 다시 실행합니다.

Sample : ServiceSample.java

```
package sample;

import oz.framework.api.Service;
import org.apache.log4j.*;
```

```
public class ServiceSample {
   public static void main(String[] args) {
      //아래 라인을 주석 처리하면 사용자가 지정하지 않은 로그가 나타나지 않습니다.
       BasicConfigurator.configure();

       // OZServer Info.
       //*
       // Daemon
       String IP = "127.0.0.1"; //서버가 설치되어 있는 호스트 컴퓨터의 IP
       int PORT = 8003; //서버가 사용하는 TCP 포트
       /*/
       // Servlet
       String URL  = "http://www.oz.com/oz/server"; //Servlet 에 접근가능한 URL
       /**/
       // User Info.
       String ID = "admin"; //default
       String PWD = "admin"; //default

       Service service = null;
       try {
          //*
          // Daemon
          service = new Service(IP, PORT, ID, PWD, false, false);
          /*/
          // Servlet
          service = new Service(URL, ID, PWD, false, false);
          /**/

         System.out.println(service.getHandlerCount());

         // garbageCollection 실행(garbageCollects)
         service.garbageCollect();
         // 서버 재시작 (restart)
         //service.restart();
         //서버 종료(serverStop)
         boolean check = true;//강제종료여부결정
         service.stop(check);

       }
       catch(Exception e)
       {
       }
    }
}
```

Class Viewer

Constructor Summary

- Viewer(String ip, int port, String id, String pw, boolean bAutoLogin, boolean useUSL)
- Viewer(String url, String id, String pw, boolean bAutoLogin, boolean useUSL)

Method Summary

- byte[] getByteArrayForm(String reportName, String categoryName)
- InputStram getForm(String reportName, String CategoryName, boolean isCompress)
- Parameter[] getUserParametersWithDefaultValue(String itemName, int itemType, String categoryName)
- HCDataModule getHCUSDM(String fileName, String categoryName)
- HCDataModule getDataModule(InputStream sdmInput)
- HCDataModule getDataModules(String odiname, String CategoryName, Parameter[] parameters, boolean doCompress, boolean forceRefresh)
- HCDataModule getDataModule(String odiName, String categoryName, Parameter[] parameters, boolean doCompress, boolean forceRefresh, String[] invalidDataset, MaxRowsOfSet[] maxRows)

Constructor Detail

Prototype	//Daemon 타입 - 오즈 서버 타입이 TCP Server인 경우 public Viewer(String ip, int port, String id, String pw, boolean bAutoLogin, boolean useUSL) //Servlet 타입 - 오즈 서버 타입이 HTTP Server인 경우 public Viewer(String url, String id, String pw, boolean bAutoLogin, boolean useUSL)	
Argument	*url*	Servlet 타입 오즈 서버의 URL ex) String url = "http://127.0.0.1/oz/server";
	ip	Daemon 타입 오즈 서버의 IP ex) String ip = "127.0.0.1";
	port	Daemon 타입 오즈 서버의 포트 번호 ex) int port = 8003;
	id	사용자 아이디 ex) String id = "admin";
	pw	사용자 패스워드 ex) String pw = "admin";
	bAutoLogin	자동 로그인 여부 ex) boolean bAutoLogin = true;
	useUSL	USL 사용 여부 ex) boolean useUSL = false;

Method Detail

■ getByteArrayForm

Prototype	public byte[] getByteArrayForm(String reportName, String categoryName) throws OZCPException	
Definition	리포트 폼의 내용을 가져옵니다.	
Argument	*reportName*	리포트 폼 이름
	categoryName	리포트의 카테고리명

■ getForm

Prototype	public InputStream getForm(String reportName, String categoryName, boolean isCompress) throws OZCPException

Definition	리포트 폼의 내용을 압축 여부를 설정하여 가져옵니다.	
Argument	*reportName*	리포트 폼 이름
	categoryName	리포트의 카테고리명
	isCompress	압축 여부

■ **getUserParametersWithDefaultValue**

Prototype	`public Parameter[] getUserParametersWithDefaultValue(String itemName, int itemType, String categoryName) throws OZCPException`	
Definition	아이템에서 사용되는 패러미터와 패러미터의 기본값을 알려줍니다.	
Argument	*itemName*	아이템 이름
	itemType	아이템의 종류 (OZR : 20001 / ODI : 10000)
	categoryName	아이템의 카테고리명
	isCompress	압축 여부

■ **getHCUSDM**

Prototype	`public HCDataModule getHCUSDM(String fileName, String categoryName) throws OZCPException`	
Definition	서버에 저장한 SDM을 데이터로 가져옵니다.	
Argument	*fileName*	리포트 폼 이름
	categoryName	리포트의 카테고리명

■ **getDataModules**

Prototype	`public HCDataModule getDataModule(InputStream sdmInput) throws OZCPException`	
Definition	SDM으로부터 데이터 레코드를 가져옵니다. FETCH 형식은 배치(batch) 타입으로, DataModule 바인딩은 2.5 방식의 Normal로 고정됩니다.	
Argument	*sdmInput*	SDM 스트림

■ **getDataModules**

Prototype	public HCDataModule getDataModule(String odiName, String categoryName, Parameter[] parameters, boolean doCompress,boolean forceRefresh) throws OZCPException public HCDataModule getDataModule(String odiName, String categoryName, Parameter[] parameters, boolean doCompress, boolean forceRefresh, String[] invalidDataset, MaxRowsOfSet[] maxRows) throws OZCPException	
Definition	선택한 데이터 레코드를 가져옵니다.	
Argument	*odiName*	데이터 소스에 대한 정보 파일명
	categoryName	데이터 소스 파일의 카테고명
	parameters	데이터 레코드를 가져오기 위한 사용자 패러미터
	doCompress	압축 전송 여부
	forceRefresh	서버의 캐시 사용 여부
	invalidDataset	데이터 소스 중 사용하지 않는 데이터셋 이름
	maxRows	데이터셋의 최대 레코드 수 정보

관련 Class

■ **Parameter(oz.dm.Parameter)**

패러미터 이름과 값에 대한 정보를 가지고 있는 클래스입니다.

- 멤버 변수
  - public String name : 패러미터 이름
  - public String value : 패러미터 값

■ **MaxRowsOfSet (oz.dm. MaxRowsOfSet)**

데이터셋의 최대 레코드 수를 제한하는 정보를 가지고 있는 클래스입니다.

- 메소드
  - SetSetName

Prototype	public void SetSetName(String v)
Definition	최대 레코드 수를 설정할 데이터셋 이름을 설정합니다.

  - GetSetName

Prototype	public String GetSetName()

Definition	최대 레코드 수가 설정되어 있는 데이터셋 이름을 가져옵니다.

- SetMaxRow

Prototype	`public void SetMaxRow(int v)`
Definition	최대 레코드 수를 설정합니다.

- GetMaxRow

Prototype	`public int GetMaxRow()`
Definition	설정되어 있는 최대 레코드 수를 가져옵니다.

■ **HCDataModule(oz.dm.hc.HCDataModule)**

HCDataSet의 모음입니다. HCDataSet은 BCDataSet이 가지고 있는 버퍼를 OZ DataModule Streaming 규약에 맞추어 파싱하여 데이터를 추출하는 역할을 합니다.

- 멤버 변수
  - public final static String PARAMETERSET_NAME = "OZParam"; : 오즈 패러미터 선언

- 메소드
  - getDataSetNames

Prototype	`public String[] getDataSetNames()`
Definition	모든 데이터셋의 이름을 리턴합니다.

  - getDataSets

Prototype	`public HCDataSet[] getDataSets(String name)`
Definition	해당 이름의 데이터셋을 가져옵니다. 데이터셋 이름은 대소문자를 구분합니다. 만일 마스터 데이터셋인 경우에는 하나를 디테일 데이터셋인 경우에는 여러 개를 가져옵니다. 해당 이름의 데이터셋이 없으면 길이(length)가 0인 배열을 반환합니다.
Argument	*name* 데이터셋 이름

  - PrintInfo

Prototype	`public void PrintInfo(PrintStream out)`
Definition	데이터와 필드 정보를 PrintStream으로 출력합니다.
Argument	*out* 출력할 PrintStream 객체

■ **HCDataSet(oz.dm.hc)**

HCDataSet은 원본 데이터의 형태와 가장 유사하게 구성되어 있는 OZ DataModule Streaming 규약에 맞는 데이터 형태입니다. 따라서 원본 데이터를 직접 얻어올 수도 있고 각각의 필드와 열에 맞는 데이터의 접근도 가능합니다.

- 멤버 변수
  - final public static int BYTEARRAY_DATA_SET = 0x1;
  - final public static int UTF_DATA_SET = 0x2;
  - final public static String BYTEARRAY_PROTOCOL_NAME = "ByteArraySet";
  - final public static String UTF_PROTOCOL_NAME = "UTFSet";

- 메소드
  - getResultSet

Prototype	`public ResultSet getResultSet()`
Definition	원본 데이터를 결과셋 형태로 가져옵니다.

Sample : ViewerSample.java

```
package sample;

import oz.framework.api.Viewer;
import oz.dm.Parameter;
import oz.dm.hc.HCDataModule;
import oz.framework.cp.OZCPException;
import oz.dm.MaxRowsOfSet;
import java.io.*;
import org.apache.log4j.*;

public class ViewerSample {
   public static void main(String[] args) {
      //아래 라인을 주석 처리하면 사용자가 지정하지 않은 로그가 나타나지 않습니다.
      BasicConfigurator.configure();

      // OZServer Info.
      //*
       // Daemon
      String IP = "127.0.0.1"; //서버가 설치되어 있는 호스트 컴퓨터의 IP
      int PORT = 8003; //서버가 사용하는 TCP 포트
      /*/
      // Servlet
      String URL  = "http://www.oz.com/oz/server"; //Servlet 에 접근가능한 URL
```

```
/**/
// User Info.
String ID = "admin"; //default
String PWD = "admin"; //default

Viewer viewer = null;
try {
   //*
   // Daemon
   viewer = new Viewer(IP, PORT, ID, PWD, false, false);
   /*/
   // Servlet
   viewer = new Viewer(URL, ID, PWD, false, false);
   /**/

   int ODI_FILE_TYPE = 10000; // ODI 파일 타입
   int OZR_FILE_TYPE = 20001; // OZR 파일 타입
   String ozrName = "test.ozr";
   String odiName = "test.odi"; //ODI 이름
   String sdmName = "test.sdm";
   String categoryName = "/"; // 카테고리 이름
   Parameter[] param = new Parameter[0]; // 패러미터
   boolean force = false; // 캐시의 리프레쉬 옵션
   boolean compress = false; // DM 의 압축 전송 여부
   String[] invalidSet = new String[0]; //사용하지 않는 데이터 셋 이름
   MaxRowsOfSet[] mx = new MaxRowsOfSet[0]; //셋들의 레코드 개수제한

   // 사용자가 서버에 설정한 SDM 에서 데이터 가져오기(getHCUSDM)
   HCDataModule usdm = viewer.getHCUSDM(sdmName, categoryName);

   // 서버에서 폼 가져오기. (getByteArrayForm)
   byte[] formBytes = viewer.getByteArrayForm(ozrName, categoryName);

   // 서버에서 폼 가져오기 (getForm)
   InputStream in = viewer.getForm(ozrName, categoryName, compress);

   // 아이템에서 사용하는 패러미터와 기본값을 반환
   //(getUserParametersWithDefaultValue)
   param = viewer.getUserParametersWithDefaultValue(odiName,
      ODI_FILE_TYPE, categoryName);

   // Input Stream 으로부터 SDM 내용을 읽어 데이터를 획득(getDataModule)
   HCDataModule sdm = viewer.getDataModule(
      new FileInputStream(sdmName));

   // 서버로부터 데이터 레코드를 획득 (getDataModules)
   HCDataModule dModule = viewer.getDataModule(
      odiName, categoryName, param, compress, force);
```

```
            // 서버로부터 데이터 레코드를 획득 (getDataModules)
            HCDataModule dModule2 = viewer.getDataModule(
                odiName, categoryName, param,compress, force, invalidSet, mx);

        }
        catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Ⅱ. 오즈 스케줄러 서버 API

- Class Program
- Class Publisher
- Class Scheduler

오즈 스케줄러 서버에 관련된 각종 정보 조회와 실시간 환경 설정 변경 기능을 사용자 애플리케이션에서 직접 제어할 수 있도록 자바 API를 제공합니다.

다음은 오즈 스케줄러 서버 API로 제공하는 주요 클래스에 대한 설명입니다.

클래스 이름	설명
Program	오즈 스케줄러에서 외부 프로그램 등록 및 관리 기능을 수행합니다.
Publisher	오즈 스케줄러에 의해 생성되는 보고서 결과 파일을 관리합니다.
Scheduler	보고서 스케줄 작업을 생성하고 결과 파일 및 스케줄러 관리 기능을 수행합니다.

오즈 스케줄러 서버 API를 사용하기 위해서는 다음과 같은 라이브러리 파일을 클래스 패스에 추가해 주어야 합니다.

라이브러리 파일명	설명
ozsfw35.jar	오즈 서버 및 Scheduler server에 접속하기 위한 파일입니다.
log4.jar	Server 내부의 클래스들을 사용할 때 Log를 남기기 위한 라이브러리 파일입니다. (API를 이용한 프로그램 실행시 classpath에 "log4.jar" 파일이 위치하여야 함)

Class Program

Constructor Summary

- **Program(String ip, int port)**

Method Summary

- **void createFolder(ServerInfo s, String folder)**
- **byte[] downloadFile(ServerInfo s, String file)**
- **FileInfo[] getExternalProgramList(ServerInfo s, String folder)**
- **void removeFiles(ServerInfo s, String folder, String[] files)**
- **void removeFolder(ServerInfo s, String folder, boolean isAll)**
- **void uploadFile(ServerInfo s, String file, byte[] b)**

Constructor Detail

Prototype	`public Program(String ip, int port) throws SchedulerException`	
Argument	*ip*	오즈 스케줄러 서버가 설치되어 있는 호스트 컴퓨터의 IP ex) String ip = "127.0.0.1";
	port	스케줄러 포트 (기본값:9521) ex) int port = 9521;

Method Detail

- **createFolder**

Prototype	`public void createFolder(ServerInfo s, String folder) throws SchedulerException`
Definition	스케줄러에서 외부 프로그램의 폴더를 생성합니다.

Argument	*s*	오즈 서버 정보
	folder	생성할 폴더명

■ **downloadFile**

Prototype	`public byte[] downloadFile(ServerInfo s, String file) throws SchedulerException`	
Definition	등록된 외부 프로그램 파일을 다운로드합니다.	
Argument	*s*	오즈 서버 정보
	file	다운로드할 파일명

■ **getExternalProgramList**

Prototype	`public FileInfo[] getExternalProgramList(ServerInfo s, String folder) throws SchedulerException`	
Definition	스케줄러에서 외부 프로그램 목록을 가져옵니다.	
Argument	*s*	오즈 서버 정보
	folder	폴더명

■ **removeFiles**

Prototype	`public void removeFiles(ServerInfo s, String folder, String[] files) throws SchedulerException`	
Definition	등록된 외부 프로그램 파일을 삭제합니다.	
Argument	*s*	오즈 서버 정보
	folder	폴더명
	files	삭제할 프로그램 파일명

■ **removeFolder**

Prototype	`public void removeFolder(ServerInfo s, String folder, boolean isAll) throws SchedulerException`	
Definition	스케줄러에서 외부 프로그램의 폴더를 삭제합니다.	
Argument	*s*	오즈 서버 정보
	folder	삭제할 폴더명
	isAll	메소드 실행 여부 true : 메소드가 실행됨(외부 프로그램 폴더 삭제됨) false : 메소드가 실행안됨(외부 프로그램 폴더 삭제안됨)

■ **uploadFile**

Prototype	`public void uploadFile(ServerInfo s, String file, byte[] b) throws SchedulerException`	
Definition	스케줄러 서버에 외부 프로그램을 등록(업로드)합니다.	
Argument	*s*	오즈 서버 정보
	file	등록할 파일명
	b	등록할 파일의 바이트 값

관련 Class

■ **SchedulerException(oz.scheduler.SchedulerException)**

오즈 스케줄러 서버에서 발생하는 Exception입니다.

■ **ServerInfo(oz.scheduler.ServerInfo)**

오즈 서버에 대한 기본적인 정보를 가지고 있는 클래스입니다.

- 메소드

▪ setIsDaemon

Prototype	`public final void setIsDaemon(boolean isDaemon)`	
Definition	오즈 서버 타입을 설정합니다.	
Argument	*isDaemon*	오즈 서버가 Daemon 타입인지 여부 • true : Daemon 타입 서버 • false : Servlet 타입 서버

▪ setIP

Prototype	`public final void setIP(String ip)`	
Definition	오즈 서버 IP를 설정합니다. Server가 Daemon 타입일 경우에만 설정합니다.	
Argument	*ip*	오즈 서버 IP

▪ setPortNo

Prototype	`public final void setPortNo(int portNo)`	
Definition	오즈 서버 Port 번호를 설정합니다. Server가 Daemon 타입일 경우에만 설정합니다.	

Argument	*portNo*	오즈 서버 Port 번호

- setURL

Prototype	`public final void setURL(String url) throws IllegalArgumentException`	
Definition	오즈 서버 URL을 설정합니다. Server가 Servlet 타입일 경우에만 설정합니다.	
Argument	*url*	오즈 서버 URL URL은 대소문자로 구분하며, 반드시 'http://'로 시작해야 합니다.

- setID

Prototype	`public final void setID(String id)`	
Definition	사용자의 ID를 설정합니다.	
Argument	*id*	사용자의 ID

- setPWD

Prototype	`public final void setPWD(String pwd)`	
Definition	사용자의 패스워드를 설정합니다.	
Argument	*pwd*	사용자의 패스워드

■ **FileInfo(oz.scheduler.FileInfo.class)**

파일 또는 폴더에 대한 기본 정보값(파일/폴더, 이름, 크기, 최종 수정시각)들을 가지고 있습니다.

- 멤버 변수
  - public boolean isDirectory : 디렉토리(폴더)인지 파일인지 구분
  - public String name : 파일명 또는 디렉토리명
  - public long size : 파일 크기 또는 디렉토리 크기
  - public long lastModified : 파일 또는 디렉토리의 최종 수정 시각, 1970년 1월 1일 00:00:00 GMT를 기준으로 일시초로 시간을 계산한 값

Sample : ProgramSample.java

```
package sample;

import oz.scheduler.ServerInfo;
import oz.scheduler.FileInfo;
import oz.framework.api.Program;
import org.apache.log4j.*;

public class ProgramSample {
   public static void main(String[] args) {
      //아래 라인을 주석 처리하면 사용자가 지정하지 않은 로그가 나타나지 않습니다.
      BasicConfigurator.configure();

      String IP = "127.0.0.1"; //스케줄러 서버가 설치되어 있는 호스트 컴퓨터의 IP
      int PORT = 9521;     //스케줄러 서버가 사용하는 TCP 포트

      Program program = null;
      try {
         program = new Program(IP, PORT);

         // 스케줄러 서버의 정보
         ServerInfo serverInfo = new ServerInfo();

         serverInfo.setID("admin");    //아이디
         serverInfo.setPWD("admin");  //패스워드
         serverInfo.setIP(IP);        //OZ Server IP 설정. daemon 일 때만 사용
         serverInfo.setIsDaemon(true);
         //OZ Server 타입 daemon, servlet 불리언 값
         serverInfo.setPortNo(8003);
         /* OZ Server Port 번호 설정.daemon 일 때만
          * s.setURL(null);//OZ Server URL 을 설정합니다.
          * URL 은 대소문자 구분하며, 반드시 //'http://' 로 시작해야
          * 합니다. Server 가 Servlet Type 일 경우에만 설정합니다.*/

         // 스케줄러 외부 프로그램 폴더 만들기(createFolder)
         String folderName = "testFolder"; //생성할 폴더 이름
         program.createFolder(serverInfo, folderName);

         // 외부 프로그램 등록하기(uploadFile)
         String fileName = "testProgram.bat"; //등록할 외부 프로그램 이름
         byte [] fileBytes = fileName.getBytes();
         program.uploadFile(serverInfo, folderName+"/"+fileName, fileBytes);

         // 등록된 외부 프로그램 내려받기(downloadFile)
         String downFileName = "testProgram.bat";//다운로드할 프로그램 파일 이름
         byte bytesArray[] =
```

```
            program.downloadFile(serverInfo, folderName+"/"+downFileName);

         // 스케줄러 외부 프로그램 목록 구하기(getExternalProgramList)
         FileInfo fileInfo[] =
            program.getExternalProgramList(serverInfo, folderName);
         for(int i=0;i<fileInfo.length;i++){
            FileInfo fi = fileInfo[i];
            System.out.println(i);
            System.out.println(" isDirectory " + fi.isDirectory);
            System.out.println(" name " + fi.name);
            System.out.println(" size " + fi.size);
            System.out.println(" lastModified " + fi.lastModified);
            System.out.println();
         }

         // 외부 프로그램 제거(removeFiles)
         // 삭제할 파일(들)의 이름을 String 배열에 넣어준다.
         String file1 = "testProgram.bat";
         String file2 = "testProgram2.bat";
         String [] files = new String[]{file1 }; //삭제할 파일들의 배열.
         program.removeFiles(serverInfo, folderName, files);

         // 외부 프로그램 폴더 제거(removeFolder)
         boolean isAll = true; //삭제 여부
         String folderNameToRemove = "testFolder";
         //삭제할 폴더명 하나만 들어간다.
         program.removeFolder(serverInfo, folderNameToRemove, isAll);

      }
      catch(Exception e)
      {
         e.printStackTrace();
      }
   }
}
```

Class Publisher

Constructor Summary

- **Publisher(String ip, int port)**

Method Summary

- **void createFolder(ServerInfo s, String folder)**
- **byte[] downloadFile(ServerInfo s, String file)**
- **FileInfo[] getPublishedFiles(ServerInfo s, String folder)**
- **void removeFiles(ServerInfo s, String folder, String[] files)**
- **void removeFolder(ServerInfo s, String folder, boolean isAll)**

Constructor Detail

Prototype	`public Publisher(String ip, int port) throws SchedulerException`	
Argument	*ip*	오즈 스케줄러 서버가 설치되어 있는 호스트 컴퓨터의 IP ex) String ip = "127.0.0.1";
	port	스케줄러 포트 (기본값:9521) ex) int port = 9521;

Method Detail

- **createFolder**

Prototype	`public void createFolder(ServerInfo s, String folder) throws SchedulerException`
Definition	스케줄러 익스포트 파일을 저장할 폴더를 생성합니다.

Argument	*s*	오즈 서버 정보
	folder	생성할 폴더명

■ **downloadFile**

Prototype	`public byte[] downloadFile(ServerInfo s, String file) throws SchedulerException`	
Definition	등록된 스케줄러 익스포트 파일을 다운로드합니다.	
Argument	*s*	오즈 서버 정보
	file	다운로드할 파일명

■ **getPublishedFiles**

Prototype	`public FileInfo[] getPublishedFiles(ServerInfo s, String folder) throws SchedulerException`	
Definition	스케줄러에서 익스포트 파일 목록을 가져옵니다.	
Argument	*s*	오즈 서버 정보
	folder	폴더명

■ **removeFiles**

Prototype	`public void removeFiles(ServerInfo s, String folder, String[] files) throws SchedulerException`	
Definition	등록된 스케줄러 익스포트 파일을 삭제합니다.	
Argument	*s*	오즈 서버 정보
	folder	삭제할 폴더명
	fiels	삭제할 파일명

■ **removeFolder**

Prototype	`public void removeFolder(ServerInfo s, String folder, boolean isAll) throws SchedulerException`	
Definition	스케줄러 익스포트 파일이 저장되는 폴더를 삭제합니다.	
Argument	*s*	오즈 서버 정보
	folder	삭제할 폴더명
	isAll	메소드 실행 여부 true : 메소드가 실행됨(외부 프로그램 폴더 삭제됨) false : 메소드가 실행안됨(외부 프로그램 폴더 삭제안됨)

관련 Class

- **ServerInfo(oz.scheduler.ServerInfo)**

  Program class의 "관련 class" 부분을 참조하시기 바랍니다.

- **FileInfo(oz. scheduler.FileInfo.class)**

  Program class의 "관련 class" 부분을 참조하시기 바랍니다.

Sample : PublisherSample.java

```
package sample;

import oz.framework.api.Publisher;
import oz.scheduler.FileInfo;
import oz.scheduler.ServerInfo;
import org.apache.log4j.*;

public class PublisherSample {
   public static void main(String[] args) {
      //아래 라인을 주석 처리하면 사용자가 지정하지 않은 로그가 나타나지 않습니다.
      BasicConfigurator.configure();

      //스케줄러 정보
      String IP = "127.0.0.1";    //스케줄러 서버가 설치되어 있는 호스트 컴퓨터의 IP
      int PORT = 9521;        //스케줄러 서버가 사용하는 TCP 포트

      Publisher publisher = null;
      try {
         publisher = new Publisher(IP, PORT);

         // 스케줄러 서버의 정보
         ServerInfo serverInfo = new ServerInfo();

         serverInfo.setID("admin");   //아이디
         serverInfo.setPWD("admin");  //패스워드
         serverInfo.setIP(IP);       //OZ Server IP 설정. daemon 일 때만 사용
         serverInfo.setIsDaemon(true);
         //OZ Server 타입 구분.daemon, servlet 불리언 값
         serverInfo.setPortNo(8003);
         /* OZ Server Port 번호 설정.daemon 일 때만
          * s.setURL(null);//OZ Server URL 을 설정합니다.
          * URL 은 대소문자 구분하며, 반드시 //'http://' 로 시작해야
          * 합니다. Server 가 Servlet Type 일 경우에만 설정합니다.*/
```

```
            // 스케줄러 생성 파일 저장 폴더 만들기(createFolder)
            String folderName = "excel"; //만들 폴더 이름
            //publisher.createFolder(serverInfo, folderName);

            // 등록된 스케줄러 생성파일을 내려받기(downloadFile)
            String fileName = "customer.xls"; //내려받을 프로그램 파일이름
            byte bytesArray[] = publisher.downloadFile(serverInfo, fileName);

            // 스케줄러 생성 파일 목록(getPublishedFiles)
            FileInfo fileInfoList[] =
               publisher.getPublishedFiles(serverInfo, folderName);
            for(int i=0; i<fileInfoList.length; i++) {
               FileInfo fi = fileInfoList[i];
               System.out.println(i);
               System.out.println(" isDirectory="+fi.isDirectory);
               System.out.println(" name="+fi.name);
               System.out.println(" size="+fi.size);
               System.out.println(" lastModified="+fi.lastModified);
               System.out.println();
            }
            // 스케줄러 생성 파일 삭제(removeFiles)
            //  삭제할 파일(들)의 이름을 String 배열에 넣어준다.
            //  customer.xls 와 orderInfo.xls 두개를 삭제하는 예시
            String file1 = "customer.xls";
            String file2 = "orderInfo.xls";
            String testFolderName = "excel";
            String [] filesToRemove = new String[]{file1, file2};
            //삭제할 파일들의 배열
            publisher.removeFiles(serverInfo, testFolderName, filesToRemove);

            // 스케줄러에서 생성되는 파일이 저장되는 폴더 제거
            boolean isAll = true;//삭제여부
            String folderNameToRemove = "excel";//삭제할 폴더 이름. 하나만 들어감.
            publisher.removeFolder(serverInfo, folderNameToRemove, isAll);

         }
         catch(Exception e)
         {
         }
      }
}
```

Class Scheduler

Constructor Summary

- scheduler(String ip, int port)

Method Summary

- void createTask(ServerInfo s, SortProperties configMap, SortProperties exportMap)
- Vector getTask(ServerInfo s)
- public TaskResult[] getTaskResult(ServerInfo s, String from, String to, String taskId) throws SchedulerException
- SortProperties[] getTaskProperties(ServerInfo info, String taskID)
- void removeTask(ServerInfo s, String task)
- String modifyTask(ServerInfo s, String taskid, SortProperties p, SortProperties exportMap)
- boolean taskPause(ServerInfo s, String task)
- boolean taskResume(ServerInfo s, String task)
- void stop(ServerInfo s, boolean waitTask)
- public boolean export(ServerInfo s, SortProperties configMap, SortProperties exportMap) throws SchedulerException
- boolean makePDF(ServerInfo s, SortProperties configMap, SortProperties exportMap)
- public boolean print(ServerInfo s, SortProperties configMap, SortProperties printMap) throws SchedulerException
- SortProperties getConfiguration(ServerInfo s)
- void modifyConfiguration(ServerInfo s, SortProperties configMap, SortProperties exportMap)

Constructor Detail

Prototype	ppublic scheduler(String ip, int port)	
Argument	*ip*	오즈 스케줄러 서버가 설치되어 있는 호스트 컴퓨터의 IP ex) String ip = "127.0.0.1";
	port	스케줄러 포트 (기본값:9521) ex) int port = 9521;

Method Detail

■ **createTask**

Prototype	public void createTask(ServerInfo s, SortProperties configMap, SortProperties exportMap) throws SchedulerException	
Definition	폼 파일을 기반으로 스케줄링 태스크를 생성합니다. ※ 참고사항 : CreateTask 함수로 여러 개의 태스크 생성 시 뷰어의 대기 큐와 상관없이 실행되는 뷰어의 개수만큼 Thread와 param 파일이 생성됩니다.	
Argument	*s*	오즈 서버 정보
	configMap	태스크를 생성할 때 사용되는 설정 옵션 설정할 수 있는 key와 설명은 아래 "Option" 부분을 참조하시기 바랍니다.
	exportMap	스케줄러에서 익스포트되는 파일의 설정 옵션 설정할 수 있는 key와 설명은 아래 "Option" 부분을 참조하시기 바랍니다.

■ **getTask**

Prototype	public Vector getTask(ServerInfo s) throws SchedulerException	
Definition	현재 생성된 스케줄링 태스크의 결과물을 가져옵니다.	
Argument	*s*	오즈 서버 정보

■ **getTaskResult**

Prototype	`public TaskResult[] getTaskResult(ServerInfo s, String from, String to, String taskId) throws SchedulerException`	
Definition	스케줄링 태스크 결과를 가져옵니다.	
Argument	*s*	오즈 서버 정보
	from	태스크 결과를 가져올 시작 시간
	to	태스크 결과를 가져올 종료 시간
	taskId	가져올 태스크 아이디

■ **getTaskProperties**

Prototype	`public SortProperties[] getTaskProperties(ServerInfo s, String taskId) throws SchedulerException`	
Definition	태스크의 속성 즉 스케줄러에서 태스크를 생성할 때 사용되는 설정 옵션(configMap)과 스케줄러에서 익스포트되는 파일의 설정 옵션(exportMap)을 가져옵니다.	
Argument	*s*	오즈 서버 정보
	taskId	속성을 가져올 태스크 아이디

■ **removeTask**

Prototype	`public void removeTask(ServerInfo s, String taskId) throws SchedulerException`	
Definition	스케줄링 태스크를 삭제합니다.	
Argument	*s*	오즈 서버 정보
	taskId	삭제할 태스크 아이디

■ **modifyTask**

Prototype	`public String modifyTask(ServerInfo s, String taskId, SortProperties configMap, SortProperties exportMap) throws SchedulerException`	
Definition	태스크 속성을 변경합니다.	
Argument	*s*	오즈 서버 정보
	taskId	속성을 변경할 태스크 아이디
	configMap	스케줄러 설정 옵션 설정할 수 있는 key와 설명은 아래 "Option" 부분을 참조하시기 바랍니다.

	exportMap	익스포트 정보 설정할 수 있는 key와 설명은 아래 "Option" 부분을 참조하시기 바랍니다.

■ **taskPause**

Prototype	public boolean taskPause(ServerInfo s, String taskId) throws SchedulerException	
Definition	스케줄링 태스크를 중지시킵니다.	
Argument	*s*	오즈 서버 정보
	taskId	중지할 태스크 아이디

■ **taskResume**

Prototype	public boolean taskResume(ServerInfo s, String taskId) throws SchedulerException	
Definition	중지된 스케줄링 태스크를 다시 실행시킵니다.	
Argument	*s*	오즈 서버 정보
	taskId	다시 실행시킬 태스크 아이디

■ **stop**

Prototype	public void stop(ServerInfo s, boolean waitTaskId) throws SchedulerException	
Definition	스케줄러를 중지시킵니다.	
Argument	*s*	오즈 서버 정보
	waitTaskId	스케줄러를 강제로 중지시킬지 여부

■ **export**

Prototype	public boolean export(ServerInfo s, SortProperties configMap, SortProperties exportMap) throws SchedulerException
Definition	뷰어 파라미터를 그대로 사용하여 익스포트한 후 익스포트 성공 여부를 반환합니다. ※ 주의사항 : 서버에 설정된 스케줄러의 환경 설정 중 "ViewType=None"이 아닌 경우에만 사용할 수 있습니다. ※ 주의사항 : 익스포트할 수 있는 파일 형식은 *.pdf, *.ozd, *.html, *.jpg, *.xls, *.doc, *.svg, *.txt, *.ppt, *.tif, *csv 파일만 가능합니다.

Argument	*s*	오즈 서버 정보
	configMap	스케줄러 설정 옵션 설정할 수 있는 key와 설명은 아래 "Option" 부분을 참조하시기 바랍니다.
	exportMap	익스포트 정보 설정할 수 있는 key와 설명은 아래 "Option" 부분을 참조하시기 바랍니다.

※ 참고사항 : export 함수를 API 함수가 아니라 COM으로 매핑하여 ASP에서 사용하는 방법은 본 매뉴얼을 "Appendix 1. SchedulerCOM 활용" 부분을 참조하시기 바랍니다.

■ **makePDF**

Prototype	`public boolean makePDF(ServerInfo s, SortProperties configMap, SortProperties exportMap) throws SchedulerException`
Definition	스케줄러의 작업 목록에 등록하지 않고 PDF 익스포트만 수행한 후 익스포트 성공 여부를 반환합니다. ※ 주의사항 : 서버에 설정된 스케줄러의 환경 설정 중 "ViewType=None"이 아닌 경우에만 사용할 수 있습니다.

Argument	*s*	오즈 서버 정보
	configMap	익스포트를 위한 폼 정보 설정할 수 있는 key와 설명은 아래 "Option" 부분을 참조하시기 바랍니다. creatTask()와 비슷하나 PDF 관련 정보만 입력하면 됩니다.
	exportMap	PDF 익스포트 정보 설정할 수 있는 key와 설명은 아래 "Option" 부분을 참조하시기 바랍니다. creatTask()와 비슷하나 PDF 관련 정보만 입력하면 됩니다.

※ 참고사항 : makePDF 함수를 API 함수가 아니라 COM으로 매핑하여 ASP에서 사용하는 방법은 본 매뉴얼을 "Appendix 1. SchedulerCOM 활용" 부분을 참조하시기 바랍니다.

■ **print**

Prototype	`public boolean print(ServerInfo s, SortProperties configMap, SortProperties printMap) throws SchedulerException`

Definition	뷰어 패러미터를 적용시켜 프린트하고 성공 여부를 반환합니다. (실제 프린트 작업의 성공 여부가 아니라 뷰어에 프린트 호출이 성공했는지 여부를 반환함) ※ 주의사항 : "task_type=viewerTag"인 경우에만 사용될 수 있으며 이외의 값이 들어오면 무시합니다. 또한 서버에 설정된 스케줄러의 환경설정 중 "ViewType=None"이 아닌 경우에만 사용할수 있습니다.	
Argument	*s*	오즈 서버 정보
	configMap	스케줄러 설정 정보 설정할 수 있는 key와 설명은 아래 "Option" 부분을 참조하시기 바랍니다.
	exportMap	출력 정보 설정할 수 있는 key와 설명은 아래 "Option" 부분을 참조하시기 바랍니다. 단, print.mode = silent로만 실행됩니다.

※ 참고사항 : print API 호출 시 태스크 작업 주기 옵션과 관계없이 즉시 한번만 프린트 작업을 호출하고 태스크는 생성되지 않으며, print API 호출 시 메일 보내기 등의 패러미터를 설정하여도 메일 관련 작업은 일어나지 않고 오직 프린트만 합니다.

■ getConfiguration

Prototype	`public SortProperties getConfiguration(ServerInfo s) throws SchedulerException`	
Definition	스케줄러 설정값을 가져옵니다.	
Argument	*s*	오즈 서버 정보

■ modifyConfiguration

Prototype	`public void modifyConfiguration(ServerInfo s, SortProperties configMap) throws SchedulerException`	
Definition	스케줄러의 설정값을 변경합니다.	
Argument	*s*	오즈 서버 정보
	configMap	스케줄러 설정 옵션 설정할 수 있는 key와 설명은 아래 "Option" 부분을 참조하시기 바랍니다.

관련 Class

■ **ServerInfo(oz.scheduler.ServerInfo)**

Program class의 "관련 class" 부분을 참조하시기 바랍니다.

■ **SortProperties (oz.util.SortProperties)**

Cache class의 "관련 class" 부분을 참조하시기 바랍니다.

■ **TaskResult(oz.scheduler.TaskResult)**

태스크 실행 결과를 나타냅니다.

- 멤버 변수
  - public String taskID: 태스크 ID
  - public String completedTime : 태스크 완료 시간
  - public int isSuccessfulCode: 태스크 성공 여부 코드 값
  - public String isSuccessful: 태스크 성공 여부
  - public String formFileName: 보고서 이름
  - public String Parameter: 패러미터
  - public String schedulingType: 스케줄 타입
  - public String exportFileList: 익스포트 파일 이름
  - public String errorMsg: 에러 발생 시 에러 메시지

Option

※ 주의사항

- 서버에 설정된 스케줄러의 환경 설정 중 "ViewType"을 "None"으로 설정한 경우에는 익스포트시에 뷰어를 사용하지 않고, 서버 API를 이용하여 익스포트 하므로, OZD 파일로만 익스포트할 수 있습니다.
- 뷰어 패러미터 중 아래 표와 같이 몇몇 패러미터는 사용자가 설정한 값은 무시되고, 항상 고정된 값으로만 설정되어 동작됩니다.

뷰어 패러미터	값
applet.mode	export
viewer.mode	export
applet.useprogressbar	false

viewer.useprogressbar	false
applet.allowmultiframe	true
viewer.allowmultiframe	true
export.mode	silent
export.confirmsave	false
information.debug	debug
applet.showerrormessage	false
viewer.showerrormessage	false

■ 기본 설정

- 리포트 이름 설정

Key	Value	설명
report_name	리포트명	태스크 생성할 리포트명 ex) setProperty("report_name", "crosstab")
category_name	카테고리명	태스크를 생성할 리포트의 카테고리명 ex) setProperty("category _name", "temp")

- 서버 데이터 모듈

Key	Value	설명
dm_server_check	"check" "null"	서버 데이터 모듈 선택 ex) setProperty("dm_server_check", "check")
dm_server_name	SDM 파일명	서버 데이터 모듈 파일명 (Repository의 odi가 있는 category에 저장된 SDM 파일명) ex) setProperty("dm_server_name", test1.sdm")
odi_name	ODI명	서버 데이터 모듈을 만들 ODI명 ex) setProperty("odi_name","testodi.odi")
odi_category_name	카테고리명	서버 데이터 모듈을 만들 ODI가 저장된 카테고리명 (루트인 경우 "/" 로 입력) ex) setProperty("odi_category_name","/test")

- 뷰어 패러미터 설정

Key	Value	설명

task_type	"viewerTag"	뷰어 패러미터 그대로 사용 여부 설정 이 옵션은 뷰어에서 사용하는 패러미터를 그대로 사용할 수 있도록 설정해 주는 옵션으로, 주로 멀티 폼을 사용하는 보고서를 익스포트할 때 사용됩니다. 사용 방법은 "SchedulerViewerTagSample.java"를 참조하시기 바랍니다. ex) setProperty("task_type ", "viewerTag")

- 외부 프로그램

Key	Value	설명
external_program_check	"check" "null"	패러미터 생성과 파일 경로의 동적 생성을 위한 외부 프로그램 체크 여부 ex) setProperty("external_program_check", "check")
external_program_command	외부 프로그램명	외부 프로그램명 ("SCH_HOME/External"에서의 상대 경로) ex) SetProperty("external_program_command", "notepad.bat")

■ 전자 메일 발송

Key	Value	설명
mail_check	"check" "null"	전자 메일 발송 여부 ex) setProperty("mail_check", "check")
mail_notify_error_check	"check" "null"	수신자에게 에러 통보 여부 선택 ex) setProperty("mail_notify_error_check", "null")
mail_recipient_to	수신자 메일 주소	전자 메일 수신자 ex) setProperty("mail_recipient_to", "gil_dong@forcs.com")
mail_recipient_cc	참조자 메일 주소	참조할 사람 메일 ex) setProperty("mail_recipient_cc", "aaa@forcs.com")
mail_recipient_bcc	숨은 참조자 메일 주소	숨은 참조할 사람 메일 ex) setProperty("mail_recipient_bcc", "bbb@forcs.com")

mail_subject	메일 제목	전자 메일 제목 ex) setProperty("mail_subject", "메일제목")
mail_text_message	메일 내용	전자 메일 내용 ex) setProperty("mail_text_message", "메일내용")
mail_html_comment	"check" "null"	전자 메일 내용에 HTML 사용 여부 ex) setProperty("mail_html_comment","check")
html_mail_content	"check" "null"	전자 메일 내용에 생성한 HTML 파일로 대체 ex) setProperty("html_mail_content","check")
mail_attach_list	첨부 파일 리스트	전자 메일에 첨부할 스케줄러의 생성된 파일 리스트 (첨부할 파일을 다수를 선택할수 있으며 '/'로 구분하여 넣습니다) ex) setProperty("mail_attach_list","excel/pdf/word")

■ **스케줄러 파일 변환**

Key	Value	설명
file_export_list	포맷 리스트	변환 파일 리스트 (다수 선택이 가능하며 '/'로 구분) ex) setProperty("file_export_list ", "xls/pdf/tif"

■ **리포트 패러미터**

Key	Value	설명
parameter_count	패러미터 수	입력 패러미터 수 ex) setProperty("parameter_count", "1")
parameter_name_1 **...** **parameter_name_n**	패러미터명	입력 패러미터명 (n : 입력 패러미터 수) ex) setProperty("parameter_name_1", "[FORM].empNo")
parameter_value_1 **...** **parameter_value_n**	패러미터 값	입력 패러미터 값 (n : 입력 패러미터 수) ex) setProperty("parameter_value_1", " 10")

※ 참고사항

- 패러미터 수 : 패러미터의 수는 폼과 각각의 ODI에서 사용하는 패러미터 수의 합을 입력 패러미터 수로 설정하고, 그 개수만큼 패러미터 이름과 해당 값을 설정해 주어야 합니다.

▪ 패러미터명

① 리포트 패러미터명은 "[FORM]." 으로 시작합니다. 예를 들어 empNo라는 패러미터는 "[FORM].empNo"로 입력하여야 합니다.

② ODI 패러미터명은 ODI명으로 시작합니다. 예를 들어 ODI명이 "testodi"이고 패러미터명이 "id"일 경우에는 "testodi.id"로 입력하여야 합니다.

■ **ODI 패러미터**

Key	Value	설명
odi_parameter_ count	패러미터 수	입력 패러미터 수 ex) setProperty("odi_parameter_count", "1")
odi_parameter_name_1 **...** **odi_parameter_name_n**	패러미터명	입력 패러미터명 (n : 입력 패러미터 수) ex) setProperty("odi_parameter_name_1", "id")
odi_parameter_value_1 **...** **odi_parameter_value_n**	패러미터 값	입력 패러미터 값 (n : 입력 패러미터 수) ex) setProperty("odi_parameter_value_1", " 10")

※ 참고사항 : ODI의 패러미터 설정은 서버 데이터 모듈을 저장할 때만 사용되며, ODI는 서버 데이터 모듈을 저장할 때 지정한 ODI를 사용합니다.

■ **스케줄 시간**

- 실행 타입

Key	Value	설명
launch_type	"once" "immediately" "periodically"	실행 타입 once : 한번 실행 immediately : 즉시 실행 periodically : 주기적 실행 ex) setProperty("launch_type", "once")

▪ launch_ type = once인 경우

Key	Value	설명

execution_year	년도	실행 시간 – 년 ex) setProperty("execution_year", "2005")
execution_month	월	실행 시간 – 월 ex) setProperty("execution_month", "12")
execution_day	일	실행 시간 – 일 ex) setProperty("execution_day", "30")
execution_hour	시간	실행 시간 – 시 ex) setProperty("execution_hour", "10")
execution_min	분	실행 시간 – 분 ex) setProperty("execution_minute", "30")

▪ launch_ type = periodically인 경우

Key	Value	설명
start_year	년도	시작 시간 – 년 ex) setProperty("start_year", "2002")
start_month	월	시작 시간 – 월 ex) setProperty("start_month", "11")
start_day	일	시작 시간 – 일 ex) setProperty("start_day", "16")
periodically_execution_day_type	"daily" "weekly" "monthly"	주기 타입 ex) setProperty("periodically_execution_day_ type", "daily")

▪ periodically_ execution_day_type = daily인 경우

Key	Value	설명
daily_type	weekday	평일 ex) setProperty("daily_type", "weekday")
daily_every_days	간격 일수	설정한 일 간격(며칠) ex) 2일 : setProperty ("daily_every_days", "2")

▪ periodically_ execution_day_type = weekly인 경우

Key	Value	설명
weekly_every_weeks	매주 간격	매주 몇번째 주 ex) setProperty("weekly_every_weeks", "2")

weekly_monday_check	"check" "null"	매주 월요일 ex) setProperty("weekly_monday_check", "check")
weekly_tuesday_check	"check" "null"	매주 화요일 ex) setProperty("weekly_tuesday_check", "check")
weekly_wednesday_check	"check" "null"	매주 수요일 ex) setProperty("weekly_wednesday_check", "check")
weekly_thursday_check	"check" "null"	매주 목요일 ex) setProperty("weekly_thursday_check", "check")
weekly_friday_check	"check" "null"	매주 금요일 ex) setProperty("weekly_friday_check", "check")
weekly_saturday_check	"check" "null"	매주 토요일 ex) setProperty("weekly_saturday_check", "check")
weekly_sunday_check	"check" "null"	매주 일요일 ex) setProperty("weekly_sunday_check", "check")

▪ periodically_ execution_day_type = monthly인 경우

Key	Value	설명
monthly_every_months	월 간격	설정한 달 간격(몇 달) ex) setProperty("monthly_every_months", "2")
monthly_type	"specific_day" , "day_of_week", "user_defined"	특정 일이나 주 ex) setProperty("monthly_type", "specific_day")
monthly_days	특정 일	매달 특정 일 ex) setProperty("monthly_days", "2")
monthly_which_week	"T1" "T2" "T3"" T4" "T5"	매달 몇 번째 주 T1 : 매달 첫번째 주 T2 : 매달 두번째 주 T3 : 매달 세번째 주 T4 : 매달 네번째 주 T5 : 매달 다섯번째 주 ex) setProperty("monthly_which_week", "T2")

monthly_which_week_day	"sunday" "monday" "tuesday" "wednesday" "thursday" "friday" "saturday"	매달 특정 요일 ex) 매달 월요일 : setProperty("monthly_which_week_day", "monday")
monthly_user_defined_days	설정할 일 리스트	설정할 일을 오름차순으로 콤마(,)로 연결하여 설정 ex) 매달 1,15일 : setProperty("monthly_user_defined_days", "1,15")

- 실행 횟수

Key	Value	설명
periodically_execution_time_ type	"once" "repeat" "user_defined"	실행 횟수(한번, 여러 번, 사용자 정의) ex) setProperty ("periodically_execution_time_type","once")

▪ periodically_ execution_time_ type = once인 경우

Key	Value	설명
once_hour	시	실행시킬 시간 – 시 ex) setProperty("once_hour", "01")
once_min	분	실행시킬 시간 – 분 ex) setProperty("once_min", "00")

▪ periodically_ execution_time_ type = repeat인 경우

Key	Value	설명
repeat_every_hours	시	주기적으로 실행 간격 – 시 ex) setProperty("repeat_every_hours", "01")
repeat_every_minutes	분	주기적으로 실행 간격 – 분 ex) setProperty("repeat_every_minutes", "04")
repeat_start_hour	시	주기적 실행 시작 시간 – 시 ex) setProperty("repeat_start_hour", "02")
repeat_start_minute	분	주기적 실행 시작 시간 – 분 ex) setProperty("repeat_start_minute", "05")

repeat_end_hour	시	주기적 실행 종료 시간 – 시 ex) setProperty("repeat_end_hour", 09")
repeat_end_minute	분	주기적 실행 종료 시간 – 분 ex) setProperty("repeat_end_minute", "35")

※ 참고사항 : 위 6가지 예제는 2시 5분에서 9시 35분 사이에 1시간 4분 간격으로 태스크 실행하라는 명령을 나타냅니다.

▪ periodically_ execution_time_ type = user_defined인 경우

Key	Value	설명
user_defined_time	설정할 시간 리스트	설정할 시간(시:분)을 오름차순으로 콤마(,)로 연결하여 설정 ex) setProperty("user_defined_time", "01:30,13:30")

■ **CSV 파일 저장**

Key	Value	설명
csv.filename	파일명	저장할 CSV 파일명 ex) setProperty("csv.filename","test.csv")
csv.pagetitle	"page"를 대체할 문자열	페이지 대체 문자열 ex) setProperty("csv.pagetitle", "<<Page>>")
csv.pageline	라인 수	페이지별 공백 라인 수 ex) setProperty("csv.pageline ", "7")
csv.pagestyle	"none" "# <page>" "# <page> ---" "--- #<page>" "<page> #"	페이지 구분 스타일 (기본값:none - 페이지 구분 없음) ex) setProperty("csv.pagestyle", "none")
csv.separator	"Tab" "Space" "Comma" 사용자정의문자	CSV 컬럼 구분자 ex) setProperty("csv.separator", "Tab")

csv.removerange	라인 영역	저장하지 않을 라인 영역 ("시작라인번호, 끝라인번호" 형식으로 설정) ex) setProperty("csv.removerage", "1, 3")
csv.exceptfirstpage	"true" "false"	첫페이지는 저장하지 않을지 여부 • true : 첫페이지 제외 • false : 첫페이지 포함(기본값) ex) setProperty("csv.exceptfirstpage", "true")
csv.savetointeger	"true" "false"	숫자 데이터를 정수로 저장할지 여부 • true : 정수로 저장 • false : 정수로 저장하지 않음(기본값) ex) setProperty("csv.savetointeger", "true")

■ **Excel 파일 저장**

Key	Value	설명
Excel.filename	파일명	저장할 엑셀 파일명 ex) setProperty("excle. filename ", "test.xls")
Excel.numberformat	셀 서식	숫자인 셀의 서식 ex) setProperty("excel.numberformat ", "#,##0.00")
Excel.savefont	폰트명 리스트	엑셀 저장 폰트 ex) setProperty("excel.savefont", "Arial, Courier")
Excel.matchmode	"columnpersheet" "paperpersheet"	엑셀 매치 모드 • columnpersheet : 페이지 열마다 별도의 Sheet로 저장(기본값) • paperpersheet : 페이지마다 별도의 Sheet 로 저장 ex) setProperty("excel.matchmode", "columnpersheet")
Excel.matchsubmode	"RowFirst" "ColumnFirst"	엑셀 매치 서브 모드(기본값:RowFirst) ex) setProperty("excel.matchsubmode", "rowfirst")
excel.removerange	라인 영역	저장하지 않을 라인 영역 ("시작라인번호, 끝라인번호" 형식으로 설정) ex) setProperty("excel.removerange", "1,3")

excel.removeoption	"FirstPageOnly" "FirstPageExcept" "AllPage"	excel.removerange가 설정되어 있을 경우 저장하지 않을 라인 영역이 적용되는 페이지 옵션 • FirstPageOnly : 첫번째 페이지만 적용 • FirstPageExcept: 모든 페이지에 적용 • AllPage : 전체 페이지에 적용(기본값) ex) setProperty("excel.removeoption", "AllPage")
excel.removeblank	"true" "false"	페이지 사이 공백 제거 여부(기본값:false) ex) setProperty("excel.removeblank", "Yes")

■ **HTML 파일 저장**

Key	Value	설명
html.filename	파일명	저장할 HTML 파일명 ex) setProperty("html.filename", "test.html")
html.imagepath	이미지 경로	Html에서 사용하는 이미지를 저장할 경로 (URL 형식으로 설정) ex) setProperty("html.imagepath","file://c:/image");
html.vertical	좌표값	페이지별 높이 여백 (단위:픽셀) 정수값으로 입력하여야 함 ex) setProperty("html.vertical","1")
html.horizontal	좌표값	페이지별 너비 여백 (단위:픽셀) 정수값으로 입력하여야 함 ex) setProperty("html.horizontal","1")
html.savebypage	"true" "false"	페이지별로 HTML 파일로 분리하여 저장할지 여부 • true : 페이지별로 저장 • false : 한 페이지로 저장(기본값) ex) setProperty("html.savebypage","true")
html.offsetx	좌표값	페이지의 x 오프셋 (단위:픽셀) 정수값으로 입력하여야 함 ex) setProperty("html.offsetx","1")
html.offsety	좌표값	페이지의 y 오프셋 (단위:픽셀) 정수값으로 입력하여야 함 ex) setProperty("html.offsety","1")

■ **Jpeg 파일 저장**

Key	Value	설명
jpg.filename	파일명	저장할 JPEG 파일명 ex) setProperty("jpg.filename", "test.jpg")

■ **OZD 파일 저장**

Key	Value	설명
ozd.filename	파일명	저장할 OZD 파일명 ex) setProperty("ozd. filename ", "test.ozd")
ozd.memoallowed	"true" "false"	메모 컴퍼넌트 사용 여부 • true : 메모 추가 사용(기본값) • false : 메모 사용하지 않음 ex) setProperty("ozd. memoallowed ", "true")
ozd.saveall	"true" "false"	모든 보고서를 저장 여부 ※ 주의사항 : Direct 익스포트시에만 적용됨 • true : 모든 보고서를 저장(기본값) • false : 현재 선택된 보고서만 저장 ex) setProperty("ozd.saveall ", "true")
ozd.password	패스워드	패스워드 ex) setProperty("ozd.password", "admin")

■ **PDF 파일 저장**

Key	Value	설명
pdf.filename	파일명	저장할 PDF 파일명 ex) setProperty("pdf.filename", "test.pdf")
pdf.saverange	저장 영역	저장 페이지 영역("시작페이지번호, 끝페이지번호" 형식으로 설정) ex) setProperty("pdf.saverange ", "1.3")
pdf.title	타이틀	PDF 문서 타이틀 ex) setProperty("pdf.title", "Report")
pdf.subject	주제	PDF 문서 주제 ex) setProperty("pdf.title", "Report")
pdf.creator	작성자	작성자 ex) setProperty("pdf.creator", "Forcs")

pdf.author	만든이	만든이 ex) setProperty("pdf.author", "Forcs")
pdf.keyword	키워드	PDF 문서 내부적으로 사용하는 키워드 ex) setProperty("pdf.keyword", "oz")
pdf.userpassword	패스워드	PDF 문서 읽기 암호 ex) setProperty("pdf.userpassword", "user")
pdf.masterpassword	패스워드	PDF 문서 편집 암호 ex) setProperty("pdf.masterpassword", "admin")
pdf.printable	인쇄 가능 여부	PDF 문서 인쇄 가능 여부(기본값:true) ex) setProperty("pdf.printable", "true")

■ **PPT 파일 저장**

Key	Value	설명
ppt.filename	파일명	저장할 PPT 파일명 ex) setProperty("ppt.filename", "test.ppt")

■ **SVG 파일 저장**

Key	Value	설명
svg.filename	파일명	저장할 SVG 파일명 ex) setProperty("svg.filename", "test.svg")

■ **Text 파일 저장**

Key	Value	설명
text.filename	파일명	저장할 Text 파일명 ex) setProperty("text.filename","test.txt")
text.pagetitle	"page"를 대체할 문자열	페이지 대체 문자열 ex) setProperty("csv.pagetitle", "<<Page>>")
text.pageline	라인 수	페이지별 공백 라인 수 ex) setProperty("text.pageline ", "7")

text.pagestyle	"none" "# <page>" "# <page> ---" "--- #<page>" "<page> #"	페이지 구분 스타일 (기본값:none - 페이지 구분 없음) ex) setProperty("text.pagestyle", "none")
text.separator	"Tab" "Space" "Comma" 사용자정의 문자	Text 컬럼 구분자 ex) setProperty("text.separator", "Tab")
text.removerage	라인 영역	저장하지 않을 라인 영역 ("시작라인번호, 끝라인번호" 형식으로 설정) ex) setProperty("text.removerage", "1, 3")
text.exceptfirstpage	"true" "false"	첫페이지는 저장하지 않을지 여부 • true : 첫페이지 제외 • false : 첫페이지 포함(기본값) ex) setProperty("text.exceptfirstpage", "true")
text.savetointeger	"true" "false"	숫자 데이터를 정수로 저장할지 여부 • true : 정수로 저장 • false : 정수로 저장하지 않음(기본값) ex) setProperty("text.savetointeger", "true")

■ **Tiff 파일 저장**

Key	Value	설명
tiff.filename	파일명	저장할 Tiff 파일명 ex) setProperty("tiff.filename", "test.tif")
tiff.encode	"G3" "G4"	Tiff 저장 인코딩 방식 • G3: fax 전송시 이용되는 G3 인코딩 방식으로 Tiff 익스포트(기본값) • G4 : fax 전송시 이용되는 G4 인코딩 방식으로 Tiff 익스포트 ex) setProperty("tiff.encode", "G3")

■ **Word 파일 저장**

Key	Value	설명
word.filename	파일명	저장할 워드 파일명 ex) setProperty("word.filename", "test.doc")

※ 참고사항

익스포트시 Value 값을 "디렉토리이름/익스포트할파일이름.파일포맷"으로 설정할 경우에는 "%스케줄러디렉토리%\Repository/디렉토리이름" 안의 파일이 익스포트 됩니다. 예를 들어, "FORCS"라는 디렉토리 안의 파일을 CSV 파일로 익스포트 하고자 할 경우 setProperty("csv.filename", "FORCS/test.csv")으로 설정하고, "%스케줄러디렉토리%\Repository/FORCS" 안에 "test.csv" 파일로 저장됩니다.

■ **modifyConfiguration의 ConfigMap**

Key	Value	설명
SchedulerPort	포트 번호	스케줄러 포트 번호 (기본값 : "9521") ex) p.setProperty("SchedulerPort", "9521");
SchedulingInfoFilePath	저장 경로	스케줄링 정보 파일 저장 경로명 ex) p.setProperty("SchedulingInfoFilePath","%SCH_HOME%/ScheduledTask");
SMTPServer	서버 주소	SMTP 서버 주소 ex) p.setProperty("SMTPServer", "mail.forcs.com");
SMTPServerProt	서버 포트 번호	SMTP 서버 포트 번호 ex) p.setProperty("SMTPServerProt", "25");
MailFrom	메일 주소	전자 메일 발송 주소 ex) p.setProperty("MailFrom", "mail@forcs.com");
TempRepositoryFilePath	저장 경로	임시 파일 저장 경로 ex) p.setProperty("TempRepositoryFilePath", "%SCH_HOME%/Te mp Repository");
RepositoryFileRootPath	저장 경로	익스포트 파일 루트 저장 경로 설정 ex) p.setProperty("RepositoryFileRootPath", "%SCH_HOME%/Repository");
ExternalProgramFilePath	저장 경로	외부 프로그램 저장 경로 설정 ex) p.setProperty("ExternalProgramFilePath", "%SCH_HOME%/External");
ErrorNotifyToSender	"true" "false"	관리자에게 에러 통보 여부 설정 • true : 통보함 • false : 통보하지 않음 ex) p.setProperty("ErrorNotifyToSender", "false");

Sample : SchedulerSample.java

```
package sample;

import oz.framework.api.*;
import oz.scheduler.TaskResult;
import oz.scheduler.ServerInfo;
import oz.scheduler.ScheduledTask;
import oz.util.SortProperties;
import java.util.Vector;
import java.io.*;
import org.apache.log4j.*;

public class SchedulerSample {
   private static String[][] mailRelatedProperties =
   {
      // 기본 설정
      {"report_name", "test.ozr"},//보고서 이름
      {"category_name","/"},     // 카테고리 이름
      {"cfg.type","new"},        // Task 의 타입 new, edit

      // 서버 데이터 모듈
      {"odi_name","testodi.odi"},
      {"odi_category_name","/test"},
      {"dm_server_check", "check"},   //서버데이터모듈
      {"dm_server_name", "test1.sdm"},//서버데이터모듈파일명

      // 외부 프로그램
      {"odi_name","testodi.odi"},   // 대상 odi 이름
      {"odi_category_name","/test"},// odi 가 저장된 카테고리이름
      {"external_program_check", "check"},
      {"external_program_command",
       "notepad.bat"},            //외부프로그램명.

      // 리포트 파라미터
      {"parameter_count", "1"},   //입력패러미터 개수
      {"parameter_name_1",
       "[FORM].empNo"},         //입력패러미터이름
      {"parameter_value_1", "10"}, //입력패러미터 값

      // ODI 파라미터
      {"odi_parameter_count", "1"},   //입력패러미터 개수
      {"odi_parameter_name_1", "id"}, //입력패러미터이름
      {"odi_parameter_value_1", "12"},//입력패러미터 값

      // 스케쥴 시간
```

```
    {"launch_type", "once"},
         //한번실행(즉시"immediately",주기적"periodically")
    {"execution_year", "2003"},//실행시간-연
    {"execution_month", "12"}, //실행시간-월
    {"execution_day", "17"},   //실행시간-일
    {"execution_hour", "10"},  //실행시간-시
    {"execution_min", "30"},   //실행시간-분
    {"periodically_execution_day_type","daily"},
         //일별(주별"weekly", 월별"monthly")
    {"daily_type", "weekday"}, //평일마다(매"specific_day")
    {"periodically_execution_time_type", "once"},
       //하루에 한번(하루에 여러번"repeat", 사용자 정의 실행"user_defined")
    {"once_hour", "01"},      //주어진 시간-시
    {"once_min", "00"},       //주어진 시간-분

    // 전자메일 발송
    {"mail_check", "check"},              //전자메일발송-선택
    {"html_mail_content", "check"},        //전자메일 메일 본문이 html 지원
    {"mail_notify_error_check", "null"},    //수신자에게 에러 통보-선택안함
    {"mail_recipient_to",
     "gil_dong@forcs.com"},               //전자메일수신자
    {"mail_recipient_cc", "abc@forcs.com"}, //참조
    {"mail_recipient_bcc", "lan@forcs.com"},//숨은 참조
    {"mail_subject", "메일제목"},          //전자메일제목
    {"mail_text_message", "메일내용"},      //전자메일내용
    {"mail_attach_list",
     "csv/excel/html/ozd/pdf/text/tiff/word/ppt/jpg/svg"},
       // 메일에 첨부하는 파일 리스트
    {"file_export_list", "csv/xls/html/ozd/pdf/txt/tif/doc/ppt/jpg/svg"}
       // Export 하는 파일 리스트
};

private static String[][] attachRelatedProperties =
    {
    // CSV 설정
    {"csv.filename ", "test.csv"},   // csv 저장할 파일 이름
    {"csv.pagetitle ", "page"},     // 페이지 타이틀 설정
    {"csv.pageline ", "7"},         // 페이지별 라인 설정
    {"csv.pagestyle", "none"},      // 페이지 스타일 설정
    {"csv.separator ", "Tab"},      // CSV 컬럼 설정
    {"csv.removerange ", "1,3"},    // CSV 저장 영역 설정
    {"csv.exceptfirstpage ", "true"},// 첫페이지 저장 예외
    {"csv.savetointeger ", "true"},  // 정수 저장여부

    // 엑셀 설정
    {"excel.filename ", "test.xls"},    // excel 저장할 파일 이름
    {"excel.numberformat ", "#,##0.00"},// 셀의 서식 설정
    {"excel.savefont", "Arial"},       // 저장 폰트 설정
```

```
{"excel.matchmode",
 "columnpersheet"},              //매치모드 설정
{"excel.matchsubmode", "rowfirst"}, //매치 서브 모드 설정
{"excel.removerange", "1,3"},      //저장 제외영역 설정
{"excel.removeoption ",
 " firstpageonly"},              //저장 제외 페이지 설정
{"excel.removeblank ", " true"},    //엑셀 저장시 페이지 사이 공백 제거

// html 설정
{"html.filename ", "test.html"}, // html 저장할 파일 이름
{"html.imagepath ",
 "file://c:/image"},           // html 의 이미지 저장경로 설정
{"html.vertical ", "1"},        // html 의 페이지별 여백 높이
{"html.horizontal ", "1"},      // html 저장시 페이지별 구분 공백 넓이
{"html.savebypage ", "true"},    // html 페이지별 분리 저장 여부
{"html.offsetx ", "1"},         //페이지 x 좌표 offset
{"html.offsety ", "1"},         //페이지 y 좌표 offset

// ozd 설정
{"ozd.filename ", "test.ozd"}, //OZD 파일 저장할 이름
{"ozd.memoallowed ", "true"},  //메모 컴포넌트 사용여부
{"ozd.password", "admin"},    // ozd 암호 설정

// pdf 설정
{"pdf.filename ", "test.pdf"},   //pdf 를 저장할 파일 이름
{"pdf.saverange ", "1.3"},      //pdf 저장할 영역 설정
{"pdf.title ", "Report"},       //pdf 타이틀 지정
{"pdf.subject ", "OZ"},         //pdf 주제 지정
{"pdf.creator ", "hong"},       //pdf 생성자 지정
{"pdf.author ", "hong"},        //pdf 만든이 설정
{"pdf.keyword ", "oz"},         //pdf 내부 키워드 설정
{"pdf.userpassword", "user"},    //pdf 열람용 암호설정
{"pdf.masterpassword ", "admin"},//pdf 수정용 암호설정
{"pdf.printprotected ", "true"}, //pdf 인쇄허용 여부 설정

// text 설정
{"text.filename ", "test.txt"},    //text 저장할 파일 이름
{"text.pagetitle ", "page"},      //페이지 타이틀 설정
{"text.pageline ", "7"},          //페이지별 라인 설정
{"text.pagestyle", "none"},       //페이지 스타일 설정
{"text.separator ", "Tab"},       //text 컬럼 설정
{"text.removerange ", "1,3"},     //text 저장 영역 설정
{"text.exceptfirstpage ", "true"}, //첫페이지 저장 예외
{"text.savetointeger ", "true"},   //정수 저장여부

// Tiff 설정
{"tiff.filename ", "test.tif"},//tiff 저장할 파일 이름
{"tiff.encode ", "G3"},        //tiff 저장 인코드 설정
```

```
    // word 설정
    {"word.filename ", "test.doc"}, //word 저장 파일 이름

    // ppt 설정
    {"ppt.filename ", "test.ppt "}, //ppt 저장 파일이름

    // jpg 설정
    {"jpg.filename ", "test.jpg "}, //jpg 저장 파일 이름

    // svg 설정
    {"svg.filename ", "test.svg"}   //svg 저장 파일 이름
};

private static String[][] schedulerRelatedProperties =
{
    {"SchedulerPort", "9521"},        //스케줄러 포트
    {"schedulerserver",
     "%SCH_HOME%/ScheduledTask"},      //스케줄링 정보 파일 저장 경로명
    {"SMTPServer", "mail.forcs.com"},  //SMTP 서버 주소
    {"SMTPServerPort", "25"},         //SMTP 서버 포트
    {"MailFrom", "kil_dong@forcs.com"},//전자메일 발송 주소
    {"TempRepositoryFilePath",
     "%SCH_HOME%/TempRepository"},     //임시 파일 저장 경로
    {"RepositoryFileRootPath",
     "%SCH_HOME%/Repository"},         //PDF 외부 라이브러리 경로
    {"ExternalProgramFilePath",
     "%SCH_HOME%/External"},           //외부 프로그램 저장 경로
    {"ErrorNotifyToSender", "false"}   //관리자에게 에러 통보
};

public static void main(String[] args) {
    //아래 라인을 주석 처리하면 사용자가 지정하지 않은 로그가 나타나지 않습니다.
    BasicConfigurator.configure();

   //스케줄러 정보
    String IP = "127.0.0.1"; //스케쥴러서버가 설치되어 있는 호스트 컴퓨터의 IP
    int PORT = 9521;        //스케쥴러서버가 사용하는 TCP 포트

    String[][] values    = null;
    Scheduler  scheduler = null;
    try {
        scheduler = new Scheduler(IP, PORT);

        // 스케쥴러 서버의 정보
        ServerInfo serverInfo = new ServerInfo();
        serverInfo.setID("admin");   //아이디
        serverInfo.setPWD("admin");  //패스워드
```

```
serverInfo.setIP(IP);       //OZ Server IP 설정. daemon 일 때만 사용
serverInfo.setIsDaemon(true);
     //OZ Server 타입 구분.daemon, servlet 불리언 값
serverInfo.setPortNo(8003);
/* OZ Server Port 번호 설정.daemon 일 때만
 * s.setURL(null);//OZ Server URL 을 설정합니다.
 * URL 은 대소문자 구분하며, 반드시 //'http://' 로 시작해야
 * 합니다. Server 가 Servlet Type 일 경우에만 설정합니다.*/

//폼 파일에서 리스트별 스케줄 태스크 생성(createTask)

//엔터프라이즈매니저 메뉴얼에 나와있는 설정 정보를 기본으로
//간단하게 태스크생성 옵션을 설정하였습니다.
//check, uncheck 의 두가지 설정을 하는 부분은 "check", "null"로설정합니다.
SortProperties props = new SortProperties();
setProperties(props, mailRelatedProperties);

SortProperties exportMap = new SortProperties();
setProperties(exportMap, attachRelatedProperties);
scheduler.createTask(serverInfo, props, exportMap);

scheduler.makePDF(serverInfo, props, exportMap);
scheduler.makePDFByPooling(serverInfo, props, exportMap);

// 스케줄러 환경 설정(modifyConfiguration)
SortProperties pro = new SortProperties();
setProperties(pro, schedulerRelatedProperties);
scheduler.modifyConfiguration(serverInfo, pro);

// 스케줄러 설정 프로퍼티 값(getConfiguration)
pro = scheduler.getConfiguration(serverInfo);
pro.list(System.out);

/*
* 스케줄링 태스크 결과(getTask)
* 스케줄러→태스크 수정
* 스케줄링 태스크 결과는 하나 이상일 수 있고, 아래
* 예제는 하나의 태스크 결과만을 설정한 것입니다.
*/
ScheduledTask sTask = new ScheduledTask();

// 태스크 결과 setting
sTask.taskID = "parameter_test.ozs";       //태스크 아이디
sTask.reportName = "/parameter_test";  //리포트 이름
sTask.schedulingTypeStr = "Periodically"; //타입
sTask.lastRunTimeStr = "";              //최종실행시간
sTask.nextRunTimeStr = "2005-06-08 19:00";//다음실행시간
sTask.status = 'W'; //상태(WAITING,RUNNING,PAUSE 의 세가지 상태 있다.)
```

```
        ScheduledTask[] taskList = scheduler.getTask(serverInfo);
        for (int i = 0; i < taskList.length; i++) {
            showTask(taskList[i]);
        }

        /*
         * 스케줄러 작업 결과(getTaskResult)
         * 스케줄러→태스크 결과
         * 선택할 태스크아이디의 개수나 태스크 결과물의 개수가 하나 이상일 수 있고, 아래
         * 예제는 하나의 태스크아이디와 결과물만을 설정 한 것입니다.
         */
        // 태스크결과물 설정
        TaskResult tr = new TaskResult();
        tr.taskID = "parameter_test.ozs";              //태스크 아이디
        tr.completedTime = "2005-06-08 19:00"; //태스크 완료 시간
        tr.isSuccessful = "true";                  //성공여부
        tr.formFileName = "/parameter_test";        //리포트 이름
        tr.paramInfo = "0{}";                      //패러미터
        tr.schedulingType = "Periodically";          //타입
        //에러메시지-발생한 에러에 따라 더 긴 에러메시지가 뜹니다. 아래 예문은
        //한줄만 표기한 것입니다.
        tr.errorMsg = "Fail to execute scheduling task with OZServer. :";
        String from = "2005-06-08"; //태스크 시작 시간
        String to = "2005-06-30";   //태스크 종료 시간
        TaskResult[] trList =
            scheduler.getTaskResult(serverInfo, from, to, tr.taskID);

        for (int i = 0; i < trList.length; i++) {
            showTaskResult(trList[i]);
        }

        String task = sTask.taskID; //태스크 아이디
        // 스케줄링 태스크 중지(taskPause)
        boolean b = scheduler.taskPause(serverInfo, task);
        // 중지된 스케줄링 태스크 다시 실행(taskResume)
        boolean bool = scheduler.taskResume(serverInfo, task);
        // 스케줄링 결과 삭제(removeTask)
        scheduler.removeTask(serverInfo, task);
        // 스케줄러 작동 중지.(stop)
        boolean waitTask = true; //작동중지여부를 boolean 으로 제어
        scheduler.stop(serverInfo, waitTask);
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
}
```

```
    private static void setProperties(SortProperties p, String[][] values) {
        for(int i=0; i<values.length; i++)
        {
            p.setProperty(values[i][0], values[i][1]);
        }
    }

    private static void showTask(ScheduledTask t) {
        System.out.println("태스크 아이디 " + t.taskID);
        System.out.println("리포트 이름 " + t.reportName);
        System.out.println("타입 " + t.schedulingTypeStr);
        System.out.println("최종실행시간 " + t.lastRunTimeStr);
        System.out.println("다음실행시간 " + t.nextRunTimeStr);
        System.out.println("상태 " + t.status);
        System.out.println("");
    }

    private static void showTaskResult(TaskResult tr) {
        System.out.println("태스크 아이디 " + tr.taskID);
        System.out.println("태스크 완료 시간 " + tr.completedTime);
        System.out.println("성공여부 " + tr.isSuccessful);
        System.out.println("리포트 이름 " + tr.formFileName);
        System.out.println("패러미터 " + tr.paramInfo);
        System.out.println("타입 " + tr.schedulingType);
        System.out.println("익스포트 파일 이름 : " + tr.exportFileList);
        System.out.println("에러메시지 " + tr.errorMsg);
        System.out.println("");

    }
}
```

Sample : SchedulerViewerTagSample.java

```
package sample;

import oz.framework.api.*;
import oz.scheduler.TaskResult;
import oz.scheduler.ServerInfo;
import oz.scheduler.ScheduledTask;
import oz.util.SortProperties;
import java.util.Vector;
import java.io.*;
import org.apache.log4j.*;
import oz.scheduler.SchedulerException;
```

```
public class SchedulerViewerTagSample {

   public static void main(String[] args) throws SchedulerException {
      //아래 라인을 주석 처리하면 사용자가 지정하지 않은 로그가 나타나지 않습니다.
      BasicConfigurator.configure();

      Scheduler  scheduler = null;
      scheduler = new Scheduler("127.0.0.1", 9521);

      // 스케쥴러 서버의 정보
      ServerInfo serverInfo = new ServerInfo();
      serverInfo.setID("admin");   //아이디
      serverInfo.setPWD("admin");  //패스워드
      serverInfo.setIsDaemon(true);
      serverInfo.setIP("127.0.0.1"); //OZ Server IP 설정. daemon 일 때만 사용
      serverInfo.setPortNo(8003);

      SortProperties config = new SortProperties();

      // 새로운 방식으로 뷰어의 태그를 직접 사용할 수 있다.
      config.setProperty("task_type", "viewerTag");

      // Task 의 타입 new, edit
      config.setProperty("cfg.type", "new");

      // 스케쥴 시간한번실행(즉시:"immediately", 주기적:"periodically")
      config.setProperty("launch_type", "immediately");

      SortProperties export = new SortProperties();

      // 뷰어 태그
      export.setProperty("connection.server", "127.0.0.1");
      export.setProperty("connection.port", "8003");
      export.setProperty("connection.reportName", "/parameter_test.ozr");
      export.setProperty("applet.mode", "export");
      export.setProperty("applet.useprogressbar", "false");
      export.setProperty("applet.allowmultiframe", "true");
      export.setProperty("connection.pcount", "2");
      export.setProperty("connection.args1", "formparam1=대");
      export.setProperty("connection.args2", "formparam2=한");
      export.setProperty("export.mode", "silent");

      // 하나의 파일로 익스포트
      export.setProperty("export.saveonefile", "true");
      export.setProperty("connection.fetchtype", "BATCH");
      export.setProperty("export.confirmsave", "false");
      export.setProperty("information.debug", "debug");
      export.setProperty("applet.showerrormessage", "false");
```

```
export.setProperty("odi.parameter_test.pcount", "2");
export.setProperty("odi.parameter_test.args1", "odiparam1=민");
export.setProperty("odi.parameter_test.args2", "odiparam2=국");
export.setProperty("odi.odinames", "parameter_test");
export.setProperty("export.format",
                  "csv/xls/html/ozd/pdf/txt/tif/doc/ppt/jpg/svg");
export.setProperty("csv.filename", "1.csv");
export.setProperty("excel.filename", "1.xls");
export.setProperty("html.filename", "1.html");
export.setProperty("ozd.filename", "1.ozd");
export.setProperty("pdf.filename", "1.pdf");
export.setProperty("text.filename", "1.txt");
export.setProperty("tiff.filename", "1.tif");
export.setProperty("word.filename", "1.doc");
export.setProperty("ppt.filename", "1.ppt");
export.setProperty("jpg.filename", "1.jpg");
export.setProperty("svg.filename", "1.svg");

// 추가로 멀티폼에 대한 정보
export.setProperty("viewer.childcount", "1");
export.setProperty("child1.connection.server", "127.0.0.1");
export.setProperty("child1.connection.port", "8003");
export.setProperty("child1.connection.reportName",
                                      "/parameter_test.ozr");
export.setProperty("child1.applet.mode", "export");
export.setProperty("child1.applet.useprogressbar", "false");
export.setProperty("child1.applet.allowmultiframe", "true");
export.setProperty("child1.export.mode", "silent");
export.setProperty("child1.connection.fetchtype", "BATCH");
export.setProperty("child1.export.confirmsave", "false");
export.setProperty("child1.information.debug", "debug");
export.setProperty("child1.applet.showerrormessage", "false");

export.setProperty("child1.connection.pcount", "2");
export.setProperty("child1.connection.args1", "formparam1=대 1");
export.setProperty("child1.connection.args2", "formparam2=한 2");

export.setProperty("child1.odi.odinames", "parameter_test");
export.setProperty("child1.odi.parameter_test.pcount", "2");
export.setProperty("child1.odi.parameter_test.args1", "odiparam1=민 3");
export.setProperty("child1.odi.parameter_test.args2", "odiparam2=국 4");

// 최초 익스포트 정보와 동일하게 설정한다.
export.setProperty("child1.export.format",
                  "csv/xls/html/ozd/pdf/txt/tif/doc/ppt/jpg/svg");
export.setProperty("child1.csv.filename", "child_1.csv");
export.setProperty("child1.excel.filename", "child_1.xls");
export.setProperty("child1.html.filename", "child_1.html");
```

```
        export.setProperty("child1.ozd.filename", "child_1.ozd");
        export.setProperty("child1.pdf.filename", "child_1.pdf");
        export.setProperty("child1.text.filename", "child_1.txt");
        export.setProperty("child1.tiff.filename", "child_1.tif");
        export.setProperty("child1.word.filename", "child_1.doc");
        export.setProperty("child1.ppt.filename", "child_1.ppt");
        export.setProperty("child1.jpg.filename", "child_1.jpg");
        export.setProperty("child1.svg.filename", "child_1.svg");

        // 이 방식은 API 로만 가능하며 EM 으로는 불가능합니다.
        String taskID = scheduler.createTask(serverInfo, config, export);

    }
}
```

Sample : SchedulerTaskResult.java

```
package sample;
import oz.framework.api.*;
import oz.scheduler.TaskResult;
import oz.scheduler.ServerInfo;
import oz.scheduler.ScheduledTask;

public class SchedulerTaskResult {

    public static void main(String[] args)
    {

        String IP = "127.0.0.1"; //scheduler server ip
        int PORT = 9521; //scheduler server port
        Scheduler scheduler = null;

        try {
            scheduler = new Scheduler(IP, PORT);

            // scheduler Server Info
            ServerInfo serverInfo = new ServerInfo();

            serverInfo.setIP("127.0.0.1"); //OZ Server IP conf. In case use of
daemon
            serverInfo.setPortNo(8003);
            serverInfo.setIsDaemon(true);//OZ Server Type. -> daemon, servlet

            serverInfo.setID("admin"); //id
            serverInfo.setPWD("admin"); //pw
```

```
        ScheduledTask[] taskList = scheduler.getTask(serverInfo);
        for (int k = 0; k < taskList.length; k++) {
            showTask(taskList[k]);

            System.out.println("\n\n");

            TaskResult tr = new TaskResult();
            tr.taskID = taskList[k].taskID;

            String from = ""; //Task Result Start time
            String to = ""; //Task Result Endt time
            boolean isComplete = false;

            TaskResult[] trList = scheduler.getTaskResult(serverInfo,
                from, to, tr.taskID);
            if (trList.length > 0) {
                for (int i = 0; i < trList.length; i++) {
                    showTaskResult(trList[i]);
                }
                isComplete = true;
            }
        }
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
}

private static void showTask(ScheduledTask t)
{
    System.out.println("TASK ID : " + t.taskID);
    System.out.println("Report Name : " + t.reportName);
    System.out.println("Type : " + t.schedulingTypeStr);
    System.out.println("Finish Execute Time : " + t.lastRunTimeStr);
    System.out.println("Next Execute Time : " + t.nextRunTimeStr);
    System.out.println("Status : " + t.status);
    System.out.println("");
}

private static void showTaskResult(TaskResult tr)
{
    System.out.println("TASK ID : " + tr.taskID);
    System.out.println("TASK FINISH TIME : " + tr.completedTime);
    System.out.println("Is Sucessed Code : " + tr.isSuccessfulCode);
    System.out.println("Is Sucessed? : " + tr.isSuccessful);
    System.out.println("Report Name : " + tr.formFileName);
```

```
        System.out.println("Parameter : " + tr.paramInfo);
        System.out.println("Type : " + tr.schedulingType);
        System.out.println("Error Message : " + tr.errorMsg);
        System.out.println("");
    }
}
```

Sample : SchedulerTaskModify.java

```
package sample;

import org.apache.log4j.*;
import oz.framework.api.*;
import oz.scheduler.ServerInfo;
import oz.scheduler.ScheduledTask;
import oz.util.SortProperties;

public class SchedulerTaskModify {

    public static void main(String[] args) {
        //아래 라인을 주석 처리하면 사용자가 지정하지 않은 로그가 나타나지 않습니다.
        BasicConfigurator.configure();

        //스케줄러 정보
        String IP = "127.0.0.1"; // 스케줄러 IP
        int PORT = 9521; // 스케줄러 Port

        Scheduler scheduler = null;
        String taskID = null;

        try {
            scheduler = new Scheduler(IP, PORT);

            // OZ Server 정보
            ServerInfo serverInfo = new ServerInfo();

            serverInfo.setIP("127.0.0.1"); //오즈 서버 IP
            serverInfo.setPortNo(8003); // 오즈 서버 Port
            serverInfo.setIsDaemon(true);// 데몬 서버

            serverInfo.setID("admin"); // 오즈 서버 ID
            serverInfo.setPWD("admin"); // 오즈 서버 Password

            // 기존에 생성한 taskID
```

```
          // task 를 수정을 하기위해서는 기존에 생성된 태스크의 task_type 은 viewerTag
가 아니어야 한다.
          taskID = "parameter_test_061031115221500_263141500.ozs";

          // 태스크의 정보를 가지고 온다.
          SortProperties[] props = scheduler.getTaskProperties(serverInfo,
taskID);

          // 기존의 txt 파일로 Export 하는 Task 에 csv 포맷으로도 export 하도록 속성을
변경.
props[0].setProperty("user_defined_time", "17:53");
          props[0].setProperty("file_export_list", "csv/txt");
          props[1].setProperty("csv.filename", "modify_task.csv");

          // 변경된 태스크 출력

System.out.println("config.....................................");
          props[0].list(System.out);


System.out.println("export.....................................");
          props[1].list(System.out);

          // 모든 태스크 목록
          ScheduledTask[] taskList = scheduler.getTask(serverInfo);
          for (int i = 0; i < taskList.length; i++) {
             if (taskID.endsWith(taskList[i].taskID)) {
                while(true) {
                   if (taskList[i].status != 'R') { //실행중이 아닌 경우
                      // 새로운 값으로 수정함
                      scheduler.modifyTask(serverInfo, taskID, props[0],
props[1]);
                      System.out.println("modify ok...");
                      break;
                   }
                   Thread.sleep(1000);
                }
             }
          }
       } catch(Exception e) {
          e.printStackTrace();
       }
    }
}
```

Sample : SchedulerPrint.java

```
package sample;

import org.apache.log4j.BasicConfigurator;
import oz.framework.api.Scheduler;
import oz.scheduler.SchedulerException;
import oz.scheduler.ServerInfo;
import oz.util.SortProperties;

public class SchedulerPrint {

   public static void main(String[] args) {
      //아래 라인을 주석 처리하면 사용자가 지정하지 않은 로그가 나타나지 않습니다.
      BasicConfigurator.configure();

      //스케줄러 정보
      String IP = "127.0.0.1"; // 스케줄러 IP
      int PORT = 9521; // 스케줄러 Port

      try {
         Scheduler scheduler = new Scheduler(IP, PORT);

         // OZ Server 정보
         ServerInfo serverInfo = new ServerInfo();
         serverInfo.setIP("127.0.0.1"); //오즈 서버 IP
         serverInfo.setPortNo(8003); // 오즈 서버 Port
         serverInfo.setIsDaemon(true); // 데몬 서버
         serverInfo.setID("admin"); // 오즈 서버 ID
         serverInfo.setPWD("admin"); // 오즈 서버 Password

         SortProperties config = new SortProperties();

         // print 기능을 사용하기 위해서는 반드시 task_type 은 viewerTag 이어야 한다.
         config.setProperty("task_type", "viewerTag");
         config.setProperty("launch_type", "immediately");

         SortProperties printMap = new SortProperties();
         printMap.setProperty("connection.server", "127.0.0.1");
         printMap.setProperty("connection.port", "8003");
         printMap.setProperty("connection.reportName",
"/parameter_test.ozr");
         printMap.setProperty("repository_agent.try_license_check", "true");
         printMap.setProperty("connection.pcount", "2");
         printMap.setProperty("connection.args1", "formparam1=대");
         printMap.setProperty("connection.args2", "formparam2=한");
         printMap.setProperty("connection.fetchtype", "BATCH");
```

```
            printMap.setProperty("odi.parameter_test.pcount", "2");
            printMap.setProperty("odi.parameter_test.args1", "odiparam1=민");
            printMap.setProperty("odi.parameter_test.args2", "odiparam2=국");
            printMap.setProperty("odi.odinames", "parameter_test");

            //print 속성(자세한 설명은 리포트 뷰어 매뉴얼 참조)
            printMap.setProperty("print.adjust", "true");
            printMap.setProperty("print.alldocument", "false");
            printMap.setProperty("print.close", "true");
            printMap.setProperty("print.collate", "true");
            printMap.setProperty("print.copies", "1");
            printMap.setProperty("print.duplex", "none");
            printMap.setProperty("print.gray", "true");
            printMap.setProperty("print.ignoreerror", "true");
            printMap.setProperty("print.lockopt", "true");
            printMap.setProperty("print.once", "false");
            printMap.setProperty("print.orientation", "default");
            printMap.setProperty("print.pageorient", "horizontal");
            printMap.setProperty("print.pagerange", "all");
            printMap.setProperty("print.printbypage", "true");
            printMap.setProperty("print.printername", "default");
            printMap.setProperty("print.scaling", "100");
            printMap.setProperty("print.size", "A4");
            printMap.setProperty("print.spoolpages", "100");
            printMap.setProperty("print.style", "normal");
            printMap.setProperty("print.usedefaultpaper", "false");
            printMap.setProperty("print.spoolpages", "100");
            printMap.setProperty("print.usedialogopt", "true");

            // 다중 보고서 출력
            printMap.setProperty("viewer.childcount", "1");
            printMap.setProperty("child1.connection.server", "127.0.0.1");
            printMap.setProperty("child1.connection.port", "8003");
            printMap.setProperty("child1.connection.reportName",
"/module_sample.ozr");
            printMap.setProperty("child1.connection.fetchtype", "BATCH");

            // 이 방식은 API 로만 가능하며 EM 으로는 불가능합니다.
            scheduler.print(serverInfo, config, printMap);
         } catch (SchedulerException se) {
            se.printStackTrace();
         }
      }

}
```

Ⅲ. 오즈 리포트 뷰어 API

- OZLauncherDll을 이용한 호출방법

본 장에서는 **OZLauncherDll**을 이용하여 다른 애플리케이션에서 오즈 뷰어를 호출하는 방법에 대해 설명합니다.

OZLauncherDll을 이용한 호출 방법

Function Summary

- **void_stdcall SetPath(LPCTSTR strpath)**
- **void_stdcall SetClassNameForJava(LPCTSTR strClassName)**
- **void_stdcall SetPackageNameForJava(LPCTSTR strPackages)**
- **void_stdcall SetMSViewerTypeForJava(int type)**
- **void_stdcall SetCommand(LPCTSTR strcommand)**
- **void_stdcall SetDataForJava(LPCTSTR strdata)**
- **BOOL_stdcall CreateOZViewer(LPCTSTR str_param, int n_type)**
- **LPCTSTR_stdcall GetResult()**
- **void_stdcall Release()**

Function Detail

- **SetPath**

Prototype	`void_stdcall SetPath(LPCTSTR strpath)`
Definition	오즈 리포트 뷰어의 경로를 설정합니다.
Argument	*strpath* 오즈 리포트 뷰어의 경로

- **SetClassNameForJava**

Prototype	`void_stdcall SetClassNameForJava(LPCTSTR strClassName)`
Definition	실행할 Applet의 Main Class를 지정합니다.(이 함수는 Applet 뷰어 호출시에만 사용합니다)
Argument	*strClassName* Main Class의 패키지 경로를 포함한 이름

Example	`SetClassNameForJava("oz.application.OZIEApplication");` `SetCommand("/string");`

■ **SetPackageNameForJava**

Prototype	`void_stdcall SetPackageNameForJava(LPCTSTR strPackages)`
Definition	포함할 패키지 목록들을 지정합니다. 이때 여러 개의 패키지를 포함할 경우 [;]로 구분하여 패키지를 나열합니다.(이 함수는 Applet 뷰어 호출시에만 사용합니다).
Argument	*strPackages* 패키지가 나열된 문자열
Example	`SetPackageNameForJava("ozapp.zip; ozviewer.jar");`

■ **SetMSViewerTypeForJava**

Prototype	`void_stdcall SetMSViewerTypeForJava(int type)`	
Definition	MS VM의 타입을 결정합니다.(이 함수는 Applet 뷰어 호출시에만 사용합니다)	
Argument	*type*	1 또는 2의 값을 가집니다. 이때 1은 Wjview를 실행하며 자바 콘솔이 보이지 않습니다. 2는 Jview를 실행하며 자바 콘솔이 보이게 됩니다.

■ **SetCommand**

Prototype	`void_stdcall SetCommand(LPCTSTR strcommand)`
Definition	뷰어 호출시 선행 처리되어야 하는 명령들을 지정합니다.
Argument	*strcommand* 선행 처리 명령을 담은 문자열
Example	`SetCommand("/locale ko/kr /mode alone /slp true /launchstring");`

■ **SetDataForJava**

Prototype	`void_stdcall SetDataForJava(LPCTSTR strdata)`
Definition	Heavy client용 데이터를 지정합니다. 이 문자열을 파싱해서 SDM을 생성하게 됩니다.(이 함수는 Applet 뷰어 호출시에만 사용합니다)
Argument	*strdata* 데이터 내용을 담고 있는 문자열

■ **CreateOZViewer**

Prototype	`BOOL_stdcall CreateOZViewer(LPCTSTR str_param, int n_type)`
Definition	오즈 리포트 뷰어를 생성하여 실행하는 함수입니다. 성공하면 true를 반환하고 실패하면 false를 반환합니다.

Argument	*str_param*	오즈 리포트 뷰어 패러미터를 설정한 문자열입니다. 패러미터들은 "₩n"으로 구분되어야 합니다.
	n_type	오즈 리포트 뷰어의 타입으로 2로 입력합니다.
Example	`CreateOZViewer("connection.server=127.0.0.1\n connection.port=8003\n connection.reportname=/ozsample.ozr\n", 2);`	

■ GetResult

Prototype	`LPCTSTR_stdcall GetResult()`
Definition	인쇄나 Export 작업을 수행하였을 경우 결과값을 알아옵니다. 이 함수는 뷰어 패러미터 중 Applet.printcommand나 Applet.exportcommand을 사용한 경우만 사용할 수 있습니다. 결과에 대한 정보가 담긴 문자열이 반환됩니다. 이때 반환되는 문자열 정보는 아래 표를 참조하시기 바랍니다.

- Return

  GetResult()에서 반환되는 값은 다음 표와 같습니다.

Export의 경우	<oz export dlg> code path file name </oz export dlg>	• code : 0 = 실패, 1 = 성공 • path : 저장 경로 • file name : 저장한 파일명
Print의 경우	<oz printer> message code report name printer name print copy print pages print range user name </oz printer>	• message : 인쇄 관련 메시지 • code : 0 = 성공, 1 = 실패 • report name : 인쇄한 보고서 이름 • printer name : 인쇄한 프린터 이름 • print copy : 인쇄 매수 • print pages : 인쇄한 페이지들 • print range : 인쇄 영역 • user name : 사용자 이름

■ Release()

Prototype	`void __stdcall Release()`
Definition	뷰어를 실행하면서 사용된 리소스를 모두 해제시키는 함수입니다. 뷰어의 사용이 끝나면 반드시 호출해 주어야 합니다.

Example

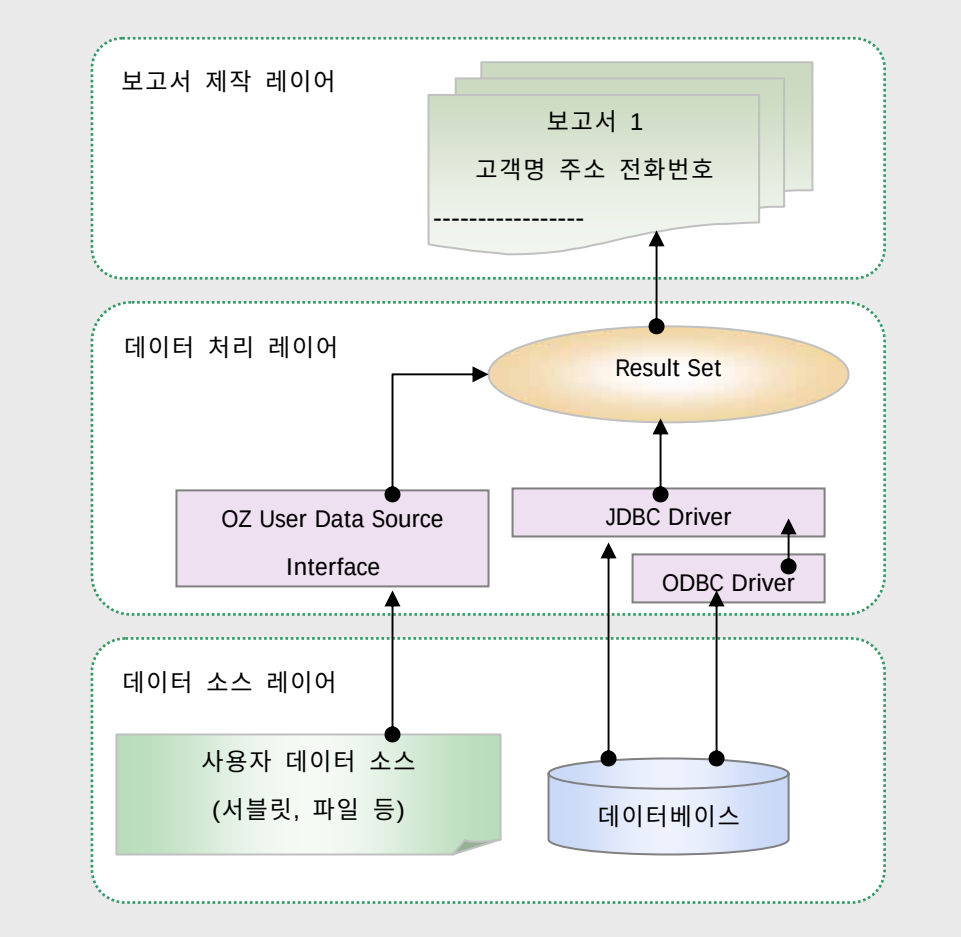
```
SetCommand("/string");
SetPath("./");
CreateOZViewer("connection.server=127.0.0.1\n
toolbar.all=true\n
information.debug=debug\n
information.bmt=true\n
connection.port=8003\n
connection.reportname=/ozsample.ozr\n
connection.compressedForm=true",1);
result = GetResult();
Release();
```

Ⅳ. User Data Store

- UDS 개요
- UDS 인터페이스
- UDS 구현

UDS 개요

UDS(User Data Store)는 오즈가 제공하는 Non-DB 데이터 연동을 위한 Java Interface로 데이터베이스에 저장되어 있지 않은 일반 텍스트 파일, CSV, XML 등의 파일 데이터는 물론 EJB, Servlet, ASP, JSP, Applet 등 사용자 프로그램의 데이터 등 다양한 형태의 외부 데이터 소스로부터 데이터셋을 생성하여 보고서를 생성할 수 있도록 합니다.

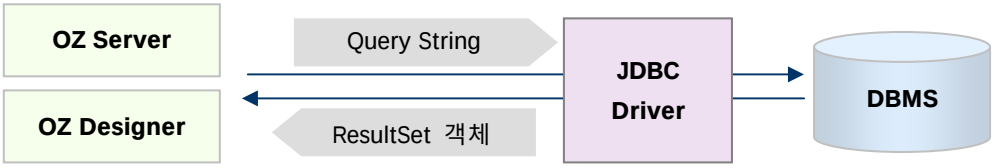


UDS 개발자는 자바를 이용하여 다양한 형태의 사용자 데이터를 받아 들이는 방법과 이 데이터를 오즈가 인식할 수 있는 ResultSet 객체로 만들어서 오즈로 넘겨주는 방법을 구현해야 합니다. 이렇게 구현된 자바 프로그램은 클래스 파일로 만들어 보고서와 연동하게 됩니다. UDS 데이터셋은 일반 SQL 데이터셋과 동일하게 사용할 수 있습니다.

오즈는 사용자 클래스로 패러미터 스트링을 전달할 수 있으며 패러미터 값에 따라 다이나믹하게 변화하는 데이터를 전달받을 수도 있습니다.

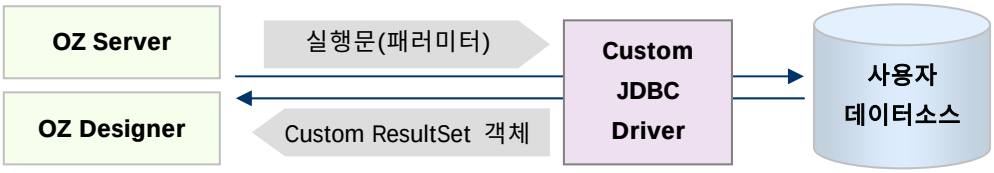
UDS 인터페이스

다음은 JDBC와 UDS 인터페이스를 비교 설명합니다.



오즈 서버는 JDBC Driver를 통해 DBMS에 접속한 후 쿼리문을 전달하고 JDBC Driver는 DBMS로부터 전달받은 쿼리 결과 데이터를 ResultSet 자바 객체에 담아 오즈 서버로 전달합니다.

이 ResultSet은 Java JDBC 사양에 따르면 java.sql.ResultSet Interface를 구현(implement)한 것이어야 합니다. 즉, DBMS의 각 벤더들은 자신의 DBMS를 지원하는 JDBC Driver를 제공할 때 java.sql.ResultSet Interface를 구현한 Concrete ResultSet 클래스를 같이 제공하는 것입니다. 이때 JDBC Driver를 제공하는 벤더가 반드시 지켜야 하는 것은 java.sql.ResultSet Interface를 따라 구현해야 하는 것입니다.



UDS(User Data Store)는 JDBC Driver의 역할을 하는 Custom Driver로 사용자가 java.sql.ResultSet interface를 따르면서 사용자 데이터 소스에 있는 데이터를 읽을 수 있는 ResultSet 클래스를 제공해 주는 것입니다.

JDBC Driver는 기능도 많고 복잡하여 구현하기가 힘듭니다. 그러나 오즈는 oz.uds.UserDataStore라는 Interface를 제공하여 사용자가 UserDataStore Interface를 따르는 클래스와 ResultSet Interface를 따르는 클래스만 작성하면 어떠한 데이터 저장소와도 연동할 수 있도록 합니다. oz.uds.UserDataStore interface는 다음과 같이 단 4개의 메소드만을 선언하고 있습니다.

```
package oz.uds;

import java.sql.*;

public interface OZUserDataStore {
   public void init() throws OZUserDataStoreException;
   public ResultSet getResultSet(String argument) throws
                                                  OZUserDataStoreException;
   public void freeResultSet(ResultSet rst);
   public void close();
}
```

각 메소드에 대한 설명은 아래 표와 같습니다.

메소드	설명
void init()	UDS가 처음 생성된 후 한번 호출되며, 사용자 데이터셋에 대한 초기화 작업을 수행합니다.
ResultSet getResultSet(String argument)	오즈 서버나 디자이너로부터 Argument(실행문)을 넘겨받아 해당 실행문에 맞는 사용자 데이터를 ResultSet 객체로 변환하여 반환합니다.
void freeResultSet(ResultSet rst)	ResultSet의 종료 작업을 수행하며, 오즈 서버나 디자이너에 getResultSet에서 받은 ResultSet을 모두 사용한 후 호출합니다.
void close()	UDS를 모두 사용한 후 마지막으로 호출되며, 사용자 데이터셋에 대한 종료 작업을 수행합니다.

※ 참고사항 : 이 중 init(), close()는 대부분의 경우 구현할 필요가 없습니다.

UDS 구현

UDS 구현에서 가장 중요한 부분은 실제 데이터를 받아오는데 필요한 getResultSet 메소드를 구현하는 것입니다.

UDS Source 구성

■ **UDS Main 구현**

오즈 서버나 디자이너에서 넘겨준 Argument에 따라 사용자 데이터를 받아 오는 부분을 구현합니다.

```
import oz.uds.*;
import java.sql.ResultSet;

public class Extdata implements OZUserDataStore{

  public void init() throws OZUserDataStoreException {
     // UDS가 처음 생성된 후 한번 호출됩니다. DB 연결 등의 초기화 작업을 수행합니다.
  }

  public ResultSet getResultSet(String argument) throws OZUserDataStoreException
  {
     // Argument에 따라서 사용자 데이터를 받고 Resultset으로 변환 후 반환합니다.
     // Argument는 OZ 디자이너에서 보고서 디자인 시에 넘기도록 설정합니다.
  }

  public void freeResultSet(ResultSet rst){
     // getResultSet에서 받아온 ResultSet을 모두 사용한 후 호출합니다.
     // 일반적으로 ResultSet을 Close하는 작업을 수행합니다.
  }

  public void close() {
     // UserDataSet을 모두 사용한 후 호출합니다.
     // 일반적으로 DB Disconnection이나 종료작업을 수행합니다.
  }
}
```

■ **사용자 ResultSet 구현**

사용자 프로그램에서 ResultSet 객체를 바로 넘겨 받을 수 있다면 별도의 ResultSet을 구현할 필요가 없습니다. 그러나 일반적으로 사용자 애플리케이션은 백터나 스트림 형태로 데이터를 넘겨주기 때문에 다음과 같이 ResultSet을 구현해야 합니다. (다음에 명시된 메소드들은 오즈에 의해 호출되는 함수를 중심으로 작성되었습니다.)

JDBC의 javasql.ResultSet Interface는 30여개가 넘는 메소드를 정의하고 있지만 오즈 UDS는 그 중 실제로 필요한 몇 개의 메소드만을 구현하고 나머지 메소드는 그대로 추가하면 됩니다.

```
import java.sql.*;
import java.math.BigDecimal;
import java.util.Calendar;
import java.util.Enumeration;

public class ResultSet_User implements ResultSet{

   public ResultSet_User( ) { }
   public boolean next() throws SQLException{
      // 다음 레코드의 인덱스 값을 반환한다.
   }

   public String getString(String columnName) throws SQLException{
      // 컬럼명으로 해당 데이터를 반환한다.
    }

    // getString외에 사용자의 데이터 타입에 따라
    //다음과 같은 Method를 추가 구현할 수 있다.
    // getBoolean(String columnName)
    // getByte(String columnName)
    // getShort(String columnName)
    // getInt(String columnName
    // getLong(String columnName
    // getFloat(String columnName)
    // getDouble(String columnName
    // getBigDecimal(String columnName)
    // getDate(String columnName)
    // getTime(String columnName)
    // getTimestamp(String columnName)
    // getBlob(String columnName)
    // getClob(String columnName)
    // getBinaryStream(String columnName)

    public ResultSetMetaData getMetaData() throws SQLException{
```

```
        // ResultSetMetaData를 반환하고
        // ResultSetMetaData의 특정 Method도 역시 필요 시 구현해야 합니다.
    }
}
```

■ **ResultSetMetaData 구현**

ResultSetMetaData에는 데이터에 대한 컬럼명과 타입 정보를 반환하기 위해 다음 세가지 메소드가 구현되어야 합니다.

```
import java.sql.SQLException;
import java.sql.Types;
import java.util.*;

public class ResultSetMetaData_User implements ResultSetMetaData{

   public ResultSetMetaData_User(String[] meta) { }

   public int getColumnCount() throws SQLException{
      // 컬럼 개수를 반환한다.
   }

   public String getColumnName(int column) throws SQLException{
      // 해당 정수값(column)번째 컬럼명을 반환한다.
   }

   public int getColumnType(int column) throws SQLException{
      // 해당 정수값(column)번째 컬럼의 타입을 반환한다
   }
}
```

UDS 예제

다음은 첫 라인에 탭으로 구분된 컬럼명이 있고, 다음 라인부터는 탭으로 구분된 데이터를 포함하고 있는 파일을 읽어 처리하는 UDS 예제입니다.

```
<UdsFile.java>

import oz.uds.*;
import java.sql.ResultSet;
import java.io.FileReader;
import java.io.IOException;
import java.util.StringTokenizer;
import java.util.Vector;
```

```
public class UdsFile implements OZUserDataStore{
   private StringBuffer Sbuffer;
   private Vector mainV;  //subV를 저장합니다.
   private Vector subV;
   private String[][] data;
   private String[] meta;

   private static final String rootFileName = "";
   private static final String mainDiv = "\n";
   private static final String subDiv = "\,";

   public void init() throws OZUserDataStoreException {
      mainV = new Vector();
   }

   public ResultSet getResultSet(String command) throws
   OZUserDataStoreException {
      FileReader in = null;
      Sbuffer = new StringBuffer();
      try{
         in = new FileReader(rootFileName + command);
         char[] buffer = new char[256];
         int n;
         while((n = in.read(buffer)) > -1){
            Sbuffer.append(new String(buffer, 0, n));
         }
      } catch(IOException e){
         e.printStackTrace();
         throw new OZUserDataStoreException(e.getMessage());
      } finally {
        try{
            if(in != null)in.close();
         } catch(IOException e){
         } finally{
            in = null;
         }
      }
      String s = Sbuffer.toString();
////////////////////////////////////////////////////////////////////////////
      StringTokenizer st1 = new StringTokenizer(s,mainDiv);
      StringTokenizer st2;
      String tmpString;
      ////////////////////meta정보//////////////////////
      if(st1.hasMoreTokens()){
         tmpString = st1.nextToken().trim();
         st2 = new StringTokenizer(tmpString,subDiv);
         subV = new Vector();
```

```
            while (st2.hasMoreTokens()){
                subV.addElement(st2.nextToken().trim());
            }
            meta = new String[subV.size()];
            subV.copyInto(meta);
        }
        ////////////////////////////////////////////
        while (st1.hasMoreTokens()) {
            tmpString = st1.nextToken().trim();
            st2 = new StringTokenizer(tmpString,subDiv);
            subV = new Vector();
            while (st2.hasMoreTokens()){
                subV.addElement(st2.nextToken().trim());
            }
            mainV.addElement(subV);
        }
        data = new String[mainV.size()][];
        for(int i = 0; i < mainV.size(); i++){
            subV = (Vector)mainV.elementAt(i);
            data[i] = new String[subV.size()];
            subV.copyInto(data[i]);
        }
        Sbuffer = null;
        mainV.clear();
        ////////////////////////확인////////////////////////////////
        if(meta == null || data == null){
            throw new OZUserDataStoreException("Data is Invalide");
        }
        return new ResultSet_File(meta, data);
    }
    public void freeResultSet(ResultSet rst){
        Sbuffer = null;
        data = null;
        meta = null;
        mainV.clear();
    }
    public void close() {
        Sbuffer = null;
        data = null;
        meta = null;
        mainV = null;
    }
}

<ResultSet_File.java>

import java.sql.*;
import java.math.BigDecimal;
```

```
import java.util.Calendar;
import java.util.Enumeration;

public class ResultSet_File implements ResultSet{

   private String[][] data;
   private String[] meta;
   private int index;
   private int size;

   public ResultSet_File(String[] meta, String[][] data) {
      this.meta = meta;
      this.data = data;
      size = data.length;
      index = -1;
   }
   public boolean next() throws SQLException{
      index++;
      if(index == size) return false;
      return true;
   }
   public String getString(String columnName) throws SQLException{
      int i = findIndex(columnName);
      if(i > -1){
         try{
            return data[index][i];
         }catch(Exception e){
            throw new SQLException("Internal Error");
         }
      }else{
         return null;
      }
   }
   private final int findIndex(String columnName){
      for(int i = 0; i < meta.length; i++){
         if(columnName.equals(meta[i])){
            return i;
         }
      }
      return -1;
   }
   public String getString(int columnIndex) throws SQLException{
      return null;
   }
   public ResultSetMetaData getMetaData() throws SQLException{
      return new ResultSetMetaData_File(meta);
   }
////////////////////////////////////////////////////////////////////////////
```

```
//아래의 메소드들은 구현할 필요없이 그대로 추가합니다.
   public boolean getBoolean(String columnName) throws SQLException{
      return false;
   }
   public byte getByte(String columnName) throws SQLException{
      return -1;
   }
   public short getShort(String columnName) throws SQLException{
      return -1;
   }
   public int getInt(String columnName) throws SQLException{
      return -1;
   }
   public long getLong(String columnName) throws SQLException{
      return -1;
   }
   public float getFloat(String columnName) throws SQLException{
      return -1;
   }
   public double getDouble(String columnName) throws SQLException{
      return -1;
   }
   public BigDecimal getBigDecimal(String columnName) throws SQLException{
      return null;
   }
   public java.sql.Date getDate(String columnName) throws SQLException{
      return null;
   }
   public java.sql.Time getTime(String columnName) throws SQLException{
      return null;
   }
   public java.sql.Timestamp getTimestamp(String columnName)  throws
   SQLException{
      return null;
   }
   public Blob getBlob(String columnName) throws SQLException{
      return null;
   }
   public Clob getClob(String columnName) throws SQLException{
      return null;
   }
   public java.io.InputStream getBinaryStream(String columnName)
      throws SQLException{
      return null;
   }
////////////////////////////////////////////////////////////////////////////
   public boolean getBoolean(int columnIndex) throws
   SQLException{return false;}
```

```
   public byte getByte(int columnIndex) throws SQLException{return -1;}
   public short getShort(int columnIndex) throws SQLException{return -1;}
   public int getInt(int columnIndex) throws SQLException{return -1;}
   public long getLong(int columnIndex) throws SQLException{return -1;}
   public float getFloat(int columnIndex) throws SQLException{return -1;}
   public double getDouble(int columnIndex) throws SQLException{return -1;}
   public BigDecimal getBigDecimal(int columnIndex) throws
      SQLException{return null;}
   public java.sql.Date getDate(int columnIndex) throws
      SQLException{return null;}
   public java.sql.Time getTime(int columnIndex) throws
      SQLException{return null;}
   public java.sql.Timestamp getTimestamp(int columnIndex) throws
      SQLException{return null;}
   public Blob getBlob(int columnIndex) throws SQLException{return null;}
   public Clob getClob(int columnIndex) throws SQLException{return null;}
   public java.io.InputStream getBinaryStream(int columnIndex)
   throws SQLException{return null;}
////////////////////////////////////////////////////////////////////////////
   public void close() throws SQLException{}
   public boolean wasNull() throws SQLException{return false;}
   public byte[] getBytes(int columnIndex) throws SQLException{return null;}
   public java.io.InputStream getAsciiStream(int columnIndex) throws
      SQLException{ return null;}
   public java.io.InputStream getUnicodeStream(int columnIndex) throws
      SQLException{return null;}
   public byte[] getBytes(String columnName) throws SQLException{return null;}
   public java.io.InputStream getAsciiStream(String columnName) throws
      SQLException{return null;}
   public java.io.InputStream getUnicodeStream(String columnName) throws
      SQLException{return null;}
   public SQLWarning getWarnings() throws SQLException{return null;}
   public void clearWarnings() throws SQLException{}
   public String getCursorName() throws SQLException{return null;}
   public Object getObject(int columnIndex) throws SQLException{return null;}
   public Object getObject(String columnName) throws SQLException{return null;}
   public int findColumn(String columnName) throws SQLException{return -1;}
   public java.io.Reader getCharacterStream(int columnIndex) throws
      SQLException{return null;}
   public java.io.Reader getCharacterStream(String columnName) throws
      SQLException{return null;}
   public BigDecimal getBigDecimal(int columnIndex, int scale) throws
      SQLException{return null;}
   public BigDecimal getBigDecimal(String columnName, int scale) throws
      SQLException{return null;}
   public boolean isBeforeFirst() throws SQLException{return false;}
   public boolean isAfterLast() throws SQLException{return false;}
   public boolean isFirst() throws SQLException{return false;}
```

```
   public boolean isLast() throws SQLException{return false;}
   public void beforeFirst() throws SQLException{}
   public void afterLast() throws SQLException{}
   public boolean first() throws SQLException{return false;}
   public boolean last() throws SQLException{return false;}
   public int getRow() throws SQLException{return -1;}
   public boolean absolute( int row ) throws SQLException{return false;}
   public boolean relative( int rows ) throws SQLException{return false;}
   public boolean previous() throws SQLException{return false;}
   public void setFetchDirection(int direction) throws SQLException{}
   public int getFetchDirection() throws SQLException{return -1;}
   public void setFetchSize(int rows) throws SQLException{}
   public int getFetchSize() throws SQLException{return -1;}
   public int getType() throws SQLException{return -1;}
   public int getConcurrency() throws SQLException{return -1;}
   public boolean rowUpdated() throws SQLException{return false;}
   public boolean rowInserted() throws SQLException{return false;}
   public boolean rowDeleted() throws SQLException{return false;}
   public Statement getStatement() throws SQLException{return null;}
   public Object getObject(int i, java.util.Map map) throws
      SQLException{return null;}
   public Ref getRef(int i) throws SQLException{return null;}
   public Array getArray(int i) throws SQLException{return null;}
   public Object getObject(String colName, java.util.Map map) throws
      SQLException{return null;}
   public Ref getRef(String colName) throws SQLException{return null;}
   public Array getArray(String colName) throws SQLException{return null;}
   public java.sql.Date getDate(int columnIndex, Calendar cal) throws
      SQLException{return null;}
   public java.sql.Date getDate(String columnName, Calendar cal) throws
      SQLException{return null;}
   public java.sql.Time getTime(int columnIndex, Calendar cal) throws
      SQLException{return null;}
   public java.sql.Time getTime(String columnName, Calendar cal) throws
      SQLException{return null;}
   public java.sql.Timestamp getTimestamp(int columnIndex, Calendar cal)
      throws SQLException{return null;}
   public java.sql.Timestamp getTimestamp(String columnName, Calendar cal)
      throws SQLException{return null;}
//////////////////////////////////////////////////////////////////////////////
   public void updateNull(int columnIndex) throws SQLException{}
   public void updateBoolean(int columnIndex, boolean x) throws SQLException{}
   public void updateByte(int columnIndex, byte x) throws SQLException{}
   public void updateShort(int columnIndex, short x) throws SQLException{}
   public void updateInt(int columnIndex, int x) throws SQLException{}
   public void updateLong(int columnIndex, long x) throws SQLException{}
   public void updateFloat(int columnIndex, float x) throws SQLException{}
   public void updateDouble(int columnIndex, double x) throws SQLException{}
```

```
public void updateBigDecimal(int columnIndex, BigDecimal x) throws
   SQLException{}
public void updateString(int columnIndex, String x) throws SQLException{}
public void updateBytes(int columnIndex, byte x[]) throws SQLException{}
public void updateDate(int columnIndex, java.sql.Date x) throws
   SQLException{}
public void updateTime(int columnIndex, java.sql.Time x) throws
   SQLException{}
public void updateTimestamp(int columnIndex, java.sql.Timestamp x)
   throws SQLException{}
public void updateAsciiStream(int columnIndex, java.io.InputStream x,
   int length) throws SQLException{}
public void updateBinaryStream(int columnIndex, java.io.InputStream x,
   int length) throws SQLException{}
public void updateCharacterStream(int columnIndex, java.io.Reader x,
   int length) throws SQLException{}
public void updateObject(int columnIndex, Object x, int scale)
   throws SQLException{}
public void updateObject(int columnIndex, Object x) throws SQLException{}
public void updateNull(String columnName) throws SQLException{}
public void updateBoolean(String columnName, boolean x) throws
   SQLException{}
public void updateByte(String columnName, byte x) throws SQLException{}
public void updateShort(String columnName, short x) throws SQLException{}
public void updateInt(String columnName, int x) throws SQLException{}
public void updateLong(String columnName, long x) throws SQLException{}
public void updateFloat(String columnName, float x) throws SQLException{}
public void updateDouble(String columnName, double x) throws SQLException{}
public void updateBigDecimal(String columnName, BigDecimal x) throws
   SQLException{}
public void updateString(String columnName, String x) throws SQLException{}
public void updateBytes(String columnName, byte x[]) throws SQLException{}
public void updateDate(String columnName, java.sql.Date x) throws
   SQLException{}
public void updateTime(String columnName, java.sql.Time x) throws
   SQLException{}
public void updateTimestamp(String columnName, java.sql.Timestamp x)
   throws SQLException{}
public void updateAsciiStream(String columnName, java.io.InputStream x,
   int length) throws SQLException{}
public void updateBinaryStream(String columnName, java.io.InputStream x,
   int length) throws SQLException{}
public void updateCharacterStream(String columnName, java.io.Reader reader,
   int length) throws SQLException{}
public void updateObject(String columnName, Object x, int scale)
   throws SQLException{}
public void updateObject(String columnName, Object x) throws SQLException{}
public void insertRow() throws SQLException{}
```

```
   public void updateRow() throws SQLException{}
   public void deleteRow() throws SQLException{}
   public void refreshRow() throws SQLException{}
   public void cancelRowUpdates() throws SQLException{}
   public void moveToInsertRow() throws SQLException{}
   public void moveToCurrentRow() throws SQLException{}
//////////////////////////////////////////////////////////////////////////
}

<ResultMetaData_File.java>

import java.sql.ResultSetMetaData;
import java.sql.SQLException;
import java.sql.Types;
import java.util.*;

public class ResultSetMetaData_File implements ResultSetMetaData{
   String[] meta;
   public ResultSetMetaData_File(String[] meta) {
      this.meta = meta;
   }
   public int getColumnCount() throws SQLException{
      return meta.length;
   }
   public String getColumnName(int column) throws SQLException{
      if(0 < column && column < getColumnCount()+1){
         return meta[column-1];
      }else{
         throw new SQLException();
      }
   }
   public int getColumnType(int column) throws SQLException{
      return Types.VARCHAR;
   }
   public int isNullable(int column) throws SQLException{return -1;}
   public int getColumnDisplaySize(int column) throws SQLException{return -1;}
   public int getPrecision(int column) throws SQLException{return -1;}
   public int getScale(int column) throws SQLException{return -1;}
   public boolean isAutoIncrement(int column) throws SQLException{
      return false;}
   public boolean isCaseSensitive(int column) throws SQLException{
      return false;}
   public boolean isSearchable(int column) throws SQLException{return false;}
   public boolean isCurrency(int column) throws SQLException{return false;}
   public boolean isSigned(int column) throws SQLException{return false;}
   public boolean isReadOnly(int column) throws SQLException{return false;}
   public boolean isWritable(int column) throws SQLException{return false;}
   public boolean isDefinitelyWritable(int column) throws SQLException{
```

```
      return false;}
   public String getColumnLabel(int column) throws SQLException{return null;}
   public String getSchemaName(int column) throws SQLException{return null;}
   public String getTableName(int column) throws SQLException{return null;}
   public String getCatalogName(int column) throws SQLException{return null;}
   public String getColumnTypeName(int column) throws SQLException{
      return null;}
   public String getColumnClassName(int column) throws SQLException{
      return null;}
}
```

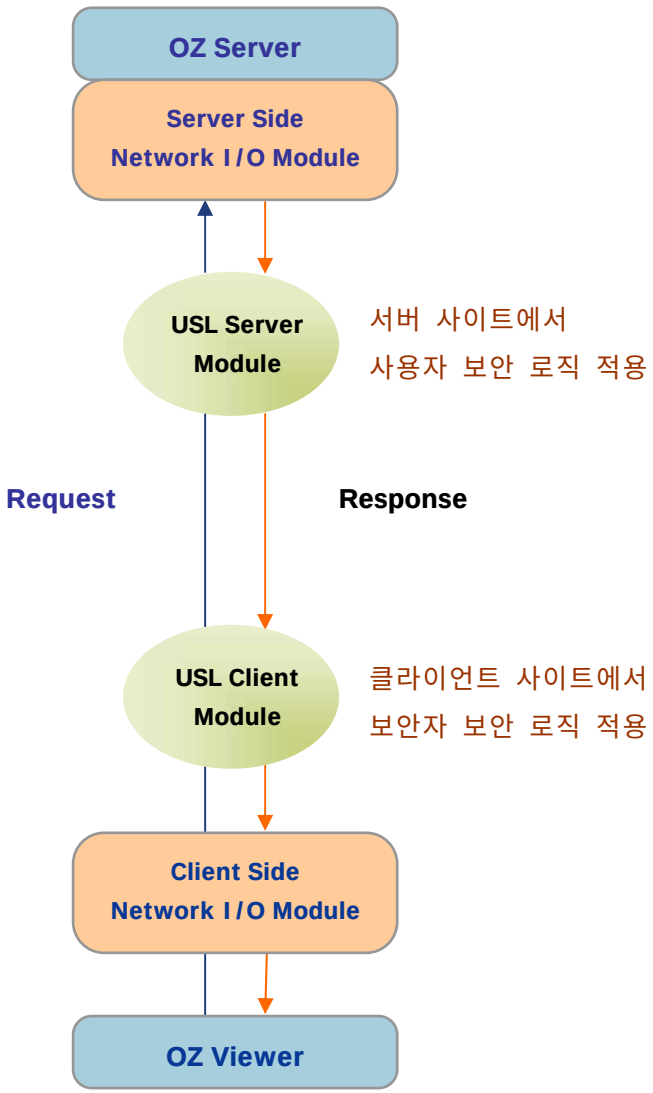
Ⅴ. User Security Logic

- USL 개요
- USL 설정
- USL 구현

USL 개요

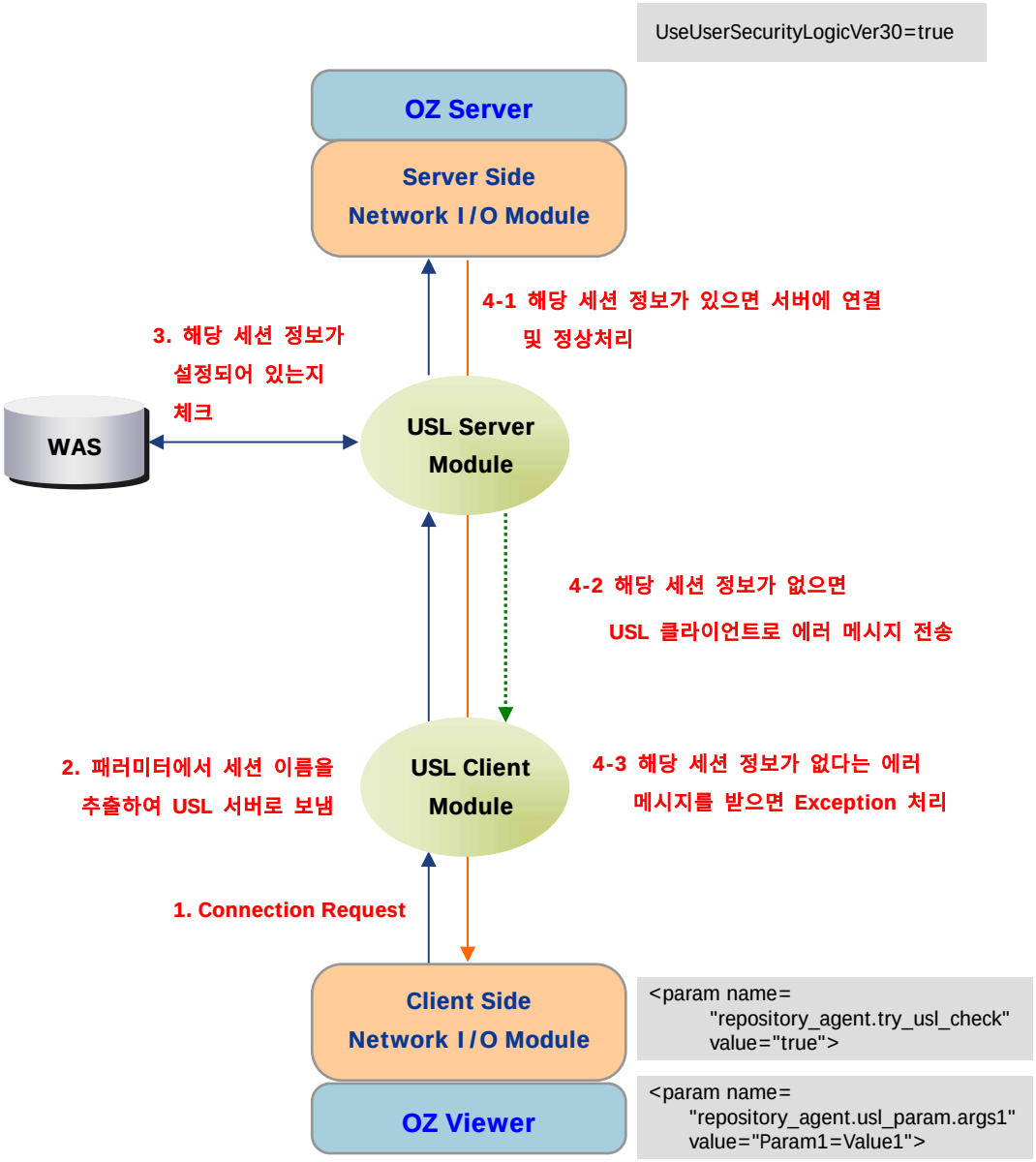
USL(User Security Logic)은 오즈 서버와 오즈 뷰어간 전송되는 데이터에 대해 사용자 보안 로직을 적용할 수 있도록 하기 위한 표준 인터페이스입니다. USL은 USL Server와 USL Client 모듈로 구성되며, 오즈 서버와 오즈 뷰어 양쪽의 네트워크 I/O 모듈 앞단에서 실행됩니다.

USL Flow



USL은 사용자 환경에서 다양하게 활용될 수 있는데 일반적으로 Web 브라우저와 WAS간의 Session 또는 Cookie 체크(오즈 서버 서블릿 타입에서만 적용 가능), Network I/O Stream에 대한 암•복호화, PKI 솔루션과의 연동을 위해 사용됩니다.

Session 체크 활용 예(기본 Session 체크 USL 제공)



USL 설정

USL을 적용하기 위해서는 오즈 서버와 오즈 뷰어를 다음과 같이 설정합니다.

오즈 서버 설정

■ **관련 파일**

- USL 관련 라이브러리 파일 : OZ_HOME/lib/ozsfw35.jar
- USL 관련 설정 파일 : OZ_HOME/conf/uslmngr.properties

■ **파일 설정값 변경**

- ozsfw35.jar 파일이 CLASSPATH에 잡혀있는지 확인합니다.
- uslmngr.properties 파일을 열어 USL 사용 여부를 설정하고, 서버 및 클라이언트 모듈의 클래스명 또는 라이브러리명을 설정합니다.

```
#
# use user security logic apply
#
UseUserSecurityLogicVer30=true
#
# default USL(Server&Client) class name (with package name)
#
OZDefault_SERVER=oz.usl.USL서버모듈 클래스명
OZDefault_CLIENT=oz.usl.USL클라이언트모듈 라이브러리명
```

  - uslmngr.properties에 UseUserSecurityLogicVer30=true로 설정하여, USL을 사용함으로 설정합니다.
  - 기본 USL 서버 및 클라이언트 모듈을 합니다.
    - OZDefault_SERVER= oz.usl.USL서버모듈 클래스명
    - OZDefault_CLIENT= oz.usl.USL클라이언트모듈 라이브러리명

    ex)

    OZDefault_SERVER=oz.usl.OZUSLServerSession

    OZDefault_CLIENT=oz.usl.OZUSLClientSession

OZ Report Viewer 설정

■ **USL 사용 여부 설정 뷰어 패러미터**

보고서 파일(OZR)를 호출할 때 USL 사용 여부를 설정하는 리포트 뷰어 패러미터인 "repository_agent.try_usl_check"를 "true"로 설정하여야 합니다.

```
<param name="repository_agent.try_usl_check" value="true">
```

※ 주의사항 : 만일 오즈 서버에서 "UseUserSecurityLogicVer30" 옵션을 "true"로 설정한 경우에는 "repository_agent.try_usl_check"를 반드시 "true"로 설정하여야 합니다.

■ **USL 패러미터 설정 뷰어 패러미터**

USL 패러미터값을 설정할 경우에는 "repository_agent.usl_param.pcount"에 USL 패러미터 개수를 설정하고, "repository_agent.usl_param_args#"를 이용하여 패러미터명과 패러미터값을 설정합니다.

※ 주의사항 : 만일 오즈 서버에서 "UseUserSecurityLogicVer30" 옵션을 "true"로 설정하고, "repository_agent.try_usl_check"을 "true"로 설정한 상태에서 USL 패러미터값을 설정하지 않으면 보고서의 기본 패러미터인 폼 패러미터가 USL 패러미터로 적용됩니다.

■ **보고서를 호출하는 HTML 파일의 예**

```
<html>
<body>
<object id="OZReportViewer" width="100%" height="100%" classid="CLSID:64DA633F-E73B-4344-83BF-48483346CD53">
    <param name="connection.servlet" value="http://127.0.0.1:8087/ozservlet/server">
    <param name="viewer.configmode" value="html">

    <param name="repository_agent.try_usl_check" value="true">
    <param name="repository_agent.usl_param_pcount" value="2">
    <param name="repository_agent.usl_param.args1" value="Param1=Value1">
    <param name="repository_agent.usl_param.args2" value="Param2=Value2">

    <param name="connection.reportname" value="sample.ozr">
</object>
</body>
</html>
```

OZ Application Viewer 설정

■ **USL 사용 여부 설정 뷰어 패러미터**

애플리케이션 폼 파일(OZA)을 호출할 때 USL 사용 여부를 설정하는 애플리케이션 뷰어 패러미터인 "repository_agent.try_usl_check"를 "true"로 설정하여야 합니다.

※ 주의사항 : 만일 오즈 서버에서 "UseUserSecurityLogicVer30" 옵션을 "true"로 설정한 경우에는 "repository_agent.try_usl_check"를 반드시 "true"로 설정하여야 합니다.

■ **USL 패러미터 설정 뷰어 패러미터**

USL 패러미터값을 설정할 경우에는 "repository_agent.usl_param.pcount"에 USL 패러미터 개수를 설정하고, "repository_agent.usl_param.args#"를 이용하여 패러미터명과 패러미터값을 설정합니다.

■ **애플리케이션 폼 파일을 호출하는 HTML 파일의 예**

```
<html>
<body>
<object id="OZApplicationViewer" width="100%" height="100%" classid="CLSID:907A00F3-7390-4EF2-931B-360546587804">
    <param name="repository_agent.type" value="FROM_OZSERVER">
    <param name="repository_agent.ozserver.servlet"
            value="http://127.0.0.1:8087/ozservlet/server">

    <param name="repository_agent.try_usl_check" value="true">
    <param name="repository_agent.usl_param.pcount" value="2">
    <param name="repository_agent.usl_param.args1" value="Param1=Value1">
    <param name="repository_agent.usl_param.args2" value="Param2=Value2">

    <param name="ozadoc.path" value="sample.oza">
</object>
</body>
</html>
```

USL 구현

USL 서버 구성

서버쪽 USL 모듈은 클라이언트쪽 USL 모듈에서 보낸 Request Stream에 대한 처리와 반대로 클라이언트로 보낼 Response Stream에 대한 처리를 담당합니다.
구현할 리스트는 다음과 같습니다.

- **USL 스트림 관리 인터페이스**

  oz.cp.OZUSLServer 추상 클래스를 상속 받아 createSecureOutputStream(), createSecureInputStream() 메소드를 구현합니다.
  클라이언트에서 온 Request Stream을 그대로 오즈 서버에 보낼 경우에는 createSecureInputStream() 메소드를 구현하지 않아도 됩니다. 마찬가지로 서버에서 받은 Response Stream을 클라이언트에 그대로 보낼 경우 createSecureOutputStream() 메소드를 구현하지 않아도 됩니다.

- **InputStream 구현**

  클라이언트로부터 받은 데이터에 어떤 처리(복호화 처리 등)를 하여 오즈 서버로 넘겨야 할 경우 InputStream 클래스를 구현해야 합니다.
  이 클래스는 java.io.InputStream 클래스를 상속받아 read() 명령을 구현합니다.
  오즈 서버에서 USL 서버로 DataInputStream을 넘겨주므로 read()가 호출될때마다 DataInputStream으로부터 데이터를 읽어서 복호화 처리 등을 한 뒤 오즈 서버로 넘겨주면 됩니다. 처음부터 생성자의 스트림으로부터 모든 데이터를 한꺼번에 가져온 후 복호화 등의 처리를 하고 read가 호출될 때마다 하나씩 반환하는 식으로 구현해도 됩니다.

- **OutputStream 구현**

  클라이언트로 보낼 데이터에 어떤 처리(암호화 처리 등)를 하고자 할 경우 OutputStream 클래스를 구현해야 합니다.
  이 클래스는 java.io.OutputStream 클래스를 상속받아 write(), flush() 명령을 구현합니다.
  오즈 서버에서 USL 서버 모듈로 DataOutputStream을 넘겨주므로 write로 들어오는 데이터를 암호화 등의 처리를 하면서 오즈 서버에서 넘겨받은 원 DataOutputStream에 써 넣어 줍니다.
  또는 write()되는 데이터를 쌓아 놓았다가 flush될 때 한꺼번에 암호화 등의 처리를 하여

원 DataOutputStream으로 보내도 됩니다. 오즈 서버는 flush를 마지막에 한번만 호출하는 것을 보장합니다.

USL 클라이언트 구성

클라이언트쪽 USL 모듈은 서버로 보낼 Request Stream에 대한 처리와 반대로 서버쪽 USL 모듈에서 보낸 Response Stream에 대한 처리를 담당합니다. 구현할 리스트는 다음과 같습니다.

- **USL 스트림 관리 인터페이스**

  OZ USL DLL프로젝트에 오즈에서 제공하는 기본 USL 소스가 제공됩니다. 서버에서 데이터를 주고 받는 스트림은 내부적인 스트림 클래스를 상속받고 이는 createSecureOutputStream(), createSecureInputStream() 메소드를 구현합니다.

  오즈 뷰어에서 온 Request Stream을 그대로 오즈 서버에 보낼 경우 createSecureOutputStream()을 구현하지 않아도 됩니다.

  마찬가지로 서버에서 온 Response Stream을 그대로 오즈 뷰어로 보낼 경우 createSecureInputStream()을 구현하지 않아도 됩니다.

  USL 클라이언트 모듈에서는 뷰어의 패러미터를 읽어서 서버에 넘겨주기도 하는데 이때 필요한 값을 서버에 넘기기 위해 createSecureOutputStream() 메소드에서 OutputStream 클래스의 writeUTF() 또는 writeINT() 등을 이용하기도 합니다.

- **InputStream 구현**

  서버로부터 받은 데이터에 어떤 처리(복호화 처리 등)를 하여 오즈 뷰어로 넘겨야 할 경우 CJDataInputStream클래스를 상속받아 구현해야 합니다.

  이 클래스는 기본적으로 오즈가 사용하는 기본적인 스트림의 메소드를 구현합니다.

  오즈 뷰어에서 USL 클라이언트로 CJDataInputStream을 넘겨주므로 read()가 호출될 때마다 CJDataInputStream으로부터 데이터를 읽어서 복호화 처리 등을 한 뒤 오즈 뷰어로 넘겨주면 됩니다.

  처음부터 생성자의 스트림으로부터 모든 데이터를 한꺼번에 가져온 후 복호화 등의 처리를 하고 read가 호출될 때마다 하나씩 반환하는 식으로 구현해도 됩니다.

- **OutputStream 구현**

  서버로 보낼 데이터에 어떤 처리(암호화 처리 등)를 하고자 할 경우 CJDataOutputStream 클래스를 구현해야 합니다.

  이 클래스는 CJDataOutputStream 클래스를 상속받아 서버로 보낼 필요한 데이터를 각각

의 구현 메소드를 통하여 전송하게 됩니다.
오즈 뷰어에서 USL 클라이언트 모듈로 CJDataOutputStream을 넘겨주며 해당 스트림을 사용하는 각각의 구현 메소드로 들어오는 데이터를 암호화 등의 처리를 하면서 오즈 뷰어에서 넘겨받은 원 CJDataOutputStream에 써 넣어 줍니다.
또는 write되는 데이터를 flush될 때 한꺼번에 암호화 등의 처리를 하여 원 CJDataOutputStream에 보내도 됩니다.
오즈 뷰어는 flush를 마지막에 한 번만 호출하는 것을 보장합니다.

USL 예제 1 - 세션 체크

USL을 이용하여 웹 브라우저와 WAS간의 사용자 세션을 체크하여 보고서 또는 애플리케이션 폼 파일 요청이 Valid한 세션 안에서 이루어지고 있는지 체크할 수 있습니다.
세션 체크를 위해서는 서버쪽에서는 OZUSLServerSession.jar를 이용하고, 클라이언트쪽에서는 OZUSLClientSession.dll 파일을 이용합니다.

■ **서버 환경 설정**

서버의 USL 환경 설정 파일인 OZ_HOME/conf/uslmngr.properties 파일을 편집기로 열어 아래와 같이 수정합니다.

```
#
# use user security logic apply
#
UseUserSecurityLogicVer30=true

#
# default USL(Server&Client) class name (with package name)
#
OZDefault_SERVER=oz.usl.OZUSLServerSession
OZDefault_CLIENT=oz.usl.OZUSLClientSession
```

■ **서버쪽 USL 모듈 구현**

OZUSLServerSession 클래스는 추상 클래스인 OZUSLServer를 상속 받습니다.
OZUSLServerSession 클래스는 클라이언트로부터 리포트 또는 애플리케이션 폼 파일을 요청 받고, WAS에 있는 OZ Servlet과 통신하여 세션을 체크합니다.
세션 체크 결과 세션이 없는 경우 사용자에게 오류 메시지를 보냅니다.

```
<<OZUSLServerSession.java>>
```

```
package oz.usl;

import java.io.*;
import java.util.Enumeration;
import javax.servlet.http.HttpSession;
import oz.datafactory.util.OZAttributeList;
import oz.cp.message.OzcmException;
import oz.framework.cp.io.OZDataInputStream;
import oz.framework.cp.io.OZDataOutputStream;

import org.apache.log4j.*;

public class OZUSLServerSession extends OZUSLServer
{
   ////////////////////////////////////////////////////////////////////
   // Non-custom section. Leave below as is.
   static protected Category cat = Category.getInstance("oz35.server");

   public OZUSLServerSession(){
      super();
   }

   ////////////////////////////////////////////////////////////////////
   // custom section. add your codes & modify

   // invoked after set httpSession & clientIP
   public void initialize(){

   }

   // invoked once per a transaction when prepare to read from client
   public InputStream createSecureInputStream(DataInputStream raw_in)
      throws OzcmException
   {
      cat.debug("OZUSLServerSession::CreateSecureInputStream");
      try{
         params_from_client.read(new OZDataInputStream(raw_in));
         for(int i=params_from_client.getLength()-1; i>=0; i--){
            cat.debug("OZUSLServerSession::CreateSecureInputStream: "
                    + params_from_client.getKey(i) + ", "
                    + params_from_client.getValue(i));
         }
      }catch(Exception e){
         cat.error("OZUSLServerSession: read params from client:" , e);
         throw new OzcmException(e.getMessage());
      }
      return null;
   }
```

```
    // invoked once per a transaction when prepare to write to client
    public OutputStream createSecureOutputStream(DataOutputStream raw_out)
        throws OzcmException
    {
        cat.debug("OZUSLServerSession::CreateSecureOutputStream");
        try{
            if(this.http_request != null){
                HttpSession http_session = http_request.getSession(false);
                if(http_session == null){
                    http_session = http_request.getSession(true);
                    http_session.setAttribute("user_id",
                                              params_from_client.get("user_id"));
                    http_session.setAttribute("user_pw",
                                              params_from_client.get("user_pw"));
                }

                for(int i = params_from_client.getLength()-1; i>=0; i--){
                    http_session.setAttribute(params_from_client.getName(i),
                                             params_from_client.getValue(i));
                }

                Enumeration e = http_session.getAttributeNames();
                for (; e.hasMoreElements(); ) {
                    String key = (String) e.nextElement();
                    String value = (String)http_session.getAttribute(key);
                    params_to_client.put(key, value);
                    cat.debug("OZUSLServerSession::http_session: "+key+", "+value);
                }
            }
            params_to_client.write(new OZDataOutputStream(raw_out));
        }catch(Exception e){
            cat.error("OZUSLServerSession: write params to client:" , e);
            throw new OzcmException(e.getMessage());
        }
        return null;
    }
}
```

※ 세션 관련 주의사항

- 세션 문제

  오즈 서버 Servlet 버전을 포팅하기 위해서는 WAS에서 /oz라는 애플리케이션을 만들어 오즈 서버(Servlet) 디렉토리를 매핑하게 되어 있는데 여기서 문제점은 루트나 특정 애플리케이션에서 설정한 세션 값이 다른 애플리케이션과 공유되지 않고 각각 따로 설정되는데 있습니다. 이는 애플리케이션 생성시 각 WEB-

INF 폴더가 생성되고 세션 값은 WEB-INF\sessions 아래에 만들어지기 때문입니다.

즉, http://localhost:8100/setcookie.jsp에서 설정한 값이 하위 폴더인 /http://localhost:8100/test/getcookie.jsp에서는 잘 나오는데 WAS에 웹 애플리케이션으로 등록된 /oz 즉, http://localhost:8100/oz/getcookie.jsp에서는 세션 값을 읽어올 수 없습니다. 마찬가지로 OZ 웹 애플리케이션에서 설정한 세션 값은 루트나 타 애플리케이션에서 읽어올 수 없습니다.

- 해결 방안

  오즈 서버를 WAS에 웹 애플리케이션으로 등록하지 않고 세션 값을 설정하는 사용자 웹 애플리케이션에 같이 포팅합니다.

  사용자 애플리케이션의 세션 값을 설정하는 Process에서 /oz 애플리케이션 아래에도 똑같은 세션을 설정하는 프로그램을 두고 똑같이 설정되도록 합니다.

  위의 문제는 Cookie를 사용할 경우에는 해당되지 않습니다.

■ **클라이언트쪽 USL 모듈 구현 – 클래스를 상속 받아 구현하는 방법**

OZUSLClientSession 클래스는 추상 클래스인 OZUSLClient를 상속 받습니다. createSecureInputStream 함수를 통해 생성된 InputStream에는 사용자 설정 내용이 포함됩니다.

```
<<OZUSLClientSession.cpp>>

#include <stdafx.h>
#include <oz/usl/OZUSLClient.h>
#include <oz/usl/OZUSLClientSession.h>
#include <common/io/PC1Encode128OutputStream.h>
#include <common/io/PC1Decode128InputStream.h>
#include <common/io/OZFuncInputStream.h>
#include <common/io/OZFuncOutputStream.h>
#include <oz/usl/OZUSLClientWrapper.h>

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

OZUSLClientSession::OZUSLClientSession() : OZUSLClient()
{
}
```

```
OZUSLClientSession::~OZUSLClientSession()
{
}

CJInputStream*    OZUSLClientSession::createSecureInputStream(CJDataInputStream*
raw_in, CJOZAttributeList& params_from_server_to_fillup, bool bInit)
{
  __super::createSecureInputStream(raw_in, params_from_server_to_fillup, bInit);
  return NULL;
}

CJOutputStream * OZUSLClientSession::createSecureOutputStream
(CJDataOutputStream* raw_out, CJOZAttributeList& params_from_client_to_send)
{
  __super::createSecureOutputStream(raw_out, params_from_client_to_send);
  return NULL;
}

void OZUSLClientSession::destory()
{
  delete this;
}
/*
 * OZUSLClient_create: initialize User Secure Logic
 *
 * this function is called every time a trasaction with OZServer occured
 *
 * @param [in] tags null terminated UTF8 OZParameter tag string array to send to
server
 * @param [in] values null terminated UTF8 OZParameter value string array to
send to server
 * @param [in] env_raw environment variable used in OZClient. just forget about
what is it.
 * however you must care about to pass env_raw to rawRead(Write)Byte(BArray)
 * functions.
 *
 * @param [in] rawReadByte function to get the encryptied raw byte from server
side USL to decrypt
 * @param [in] rawReadBArray function to get the encryptied raw byte array from
server side USL to decrypt
 * @param [in] rawAvailable function to get the tempory readable input buffer
size
 * @param [in] rawReadClose function to close raw inputstream
 *
 * @param [in] rawWriteByte function to get the encryptied raw byte to send to
server side USL
 * @param [in] rawWriteBArray function to get the encryptied raw byte array to
```

```
send to server side USL
 * @param [in] rawFlush function to flush raw outputstream
 * @param [in] rawWriteClose function to close raw outputstream
 *
 * @param [in] rawGetLastErrorMessage function to get error message from raw
in/out stream
 *
 * @param [out] do_read_hook whether do input stream wrappering with
OZUSLClient_read or not
 * @param [out] do_write_hook whether do output stream wrappering with
OZUSLClient_write or not
 * @param [out] env the state variable if needs.
 * state variable is passed to all USL functions
 * to cover stateless function call interface.
 *
 * @return returns >= 0 if success returns returns < 0 if error.
 * if got error return, OZClient will call OZUSLClient_getLastErrorMessage
 * to get detailed error message.
 */

extern "C" __declspec(dllexport) int OZUSLClient_create
(
  byte* params_bytes, int params_bytes_len,
  void* env_raw,
  int (__cdecl *rawReadByte)(void* env_raw, byte* p_value, int *read_len),
  int (__cdecl *rawReadBArray)(void* env_raw, byte* array_value, int len, int
*read_len),
  int (__cdecl *rawAvailable)(void* env_raw, int *available_len),
  int (__cdecl *rawReadClose)(void* env_raw, BOOL do_not_close_raw_in),
  int (__cdecl *rawWriteByte)(void* env_raw, byte value),
  int (__cdecl *rawWriteBArray)(void* env_raw, byte* array_value, int len),
  int (__cdecl *rawFlush)(void* env_raw),
  int (__cdecl *rawWriteClose)(void* env_raw),

  byte* (__cdecl *rawGetLastErrorMessage)(void* env_raw),

  void** env
)
{
  TRACE(_T("OZUSLClient_create\n"));
  OZUSLClientWrapper *uslw = NULL;
  uslw = new OZUSLClientWrapper();
  uslw->raw_in = new CJDataInputStream(new OZFuncInputStream(env_raw,
                    uslw, rawReadByte, rawReadBArray, rawAvailable,
                    rawReadClose, rawGetLastErrorMessage), TRUE);
  uslw->raw_out = new CJDataOutputStream(new OZFuncOutputStream(env_raw,
                    rawWriteByte, rawWriteBArray, rawFlush,
                    rawWriteClose, rawGetLastErrorMessage), TRUE);
```

```
  uslw->ozusl = new OZUSLClientSession();

  CJOZAttributeList attrs;
  CJByteArrayInputStream bin((char*)params_bytes, params_bytes_len, FALSE);
  CJDataInputStream din(&bin, FALSE);
  attrs.read(din);
  uslw->ozusl->setServerIP(attrs.get(_T("OZ_SERVER_IP")));
  uslw->ozusl->setServerURL(attrs.get(_T("OZ_SERVER_URL")));
  *env = uslw;
  return OZUSLCLIENT_INTERFACE_VERSION; // all ok.
}
```

■ **클라이언트쪽 USL 모듈 구현 – 오즈 스크립트로 구현하는 방법**

오즈 스크립트(getUSLServerParam)를 이용하여 서버네서 설정된 USL패러미터 값을 가져올 수 있습니다.

다음은 보고서에서 서버에 설정된 USL패러미터 값을 가져오는 예제입니다.

- 리포트 디자이너를 구동한 후 [파일]-[새로 만들기] 메뉴를 선택하거나 툴바의 새로 만들기 아이콘()을 클릭하여 새 보고서를 생성합니다.

- 추가된 리포트 위에 [추가]-[컴포넌트]-[밴드]-[타이틀 밴드] 메뉴를 선택하거나 툴바의 타이틀 밴드 아이콘()을 클릭하여 타이틀 밴드 하나를 추가합니다.

- 타이틀 밴드 위에 [추가]-[컴포넌트]-[라벨]-[라벨] 메뉴를 선택하거나 툴바의 라벨 아이콘()을 클릭하여 라벨을 하나 추가합니다.

- 라벨을 오른쪽 마우스로 선택하면 나타나는 [스크립트 편집] 메뉴를 선택 한 후 아래 스크립트 창에 다음과 같은 스크립트를 추가합니다.

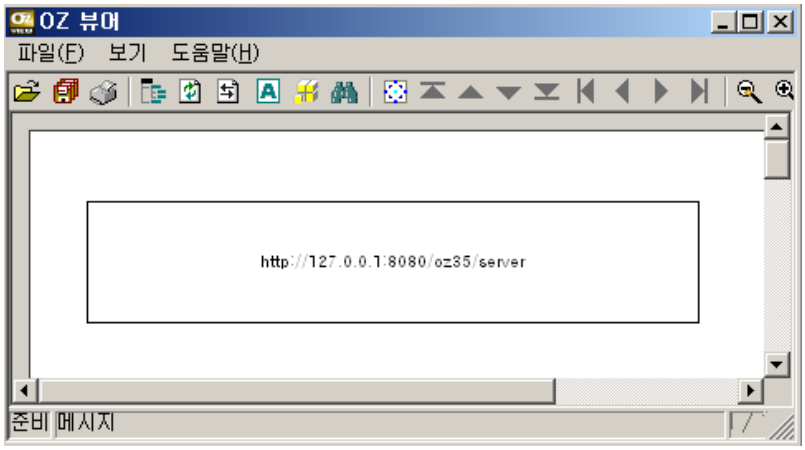
```
string str;
str = getUSLServerParam("OZUSL_SERVER_URL");
// name에 해당하는 USL패러미터값이 OZUSL_SERVER_URL 일 경우
setattr("caption",str);
write("str"); // 디버그인포(뷰어에서 CTRL+Z)에서 확인할 경우
```

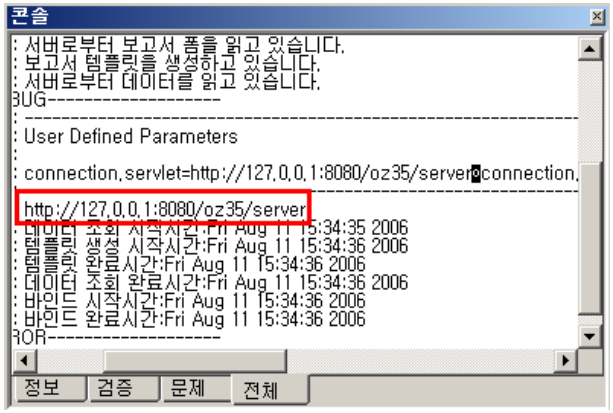
- 메뉴바에서 [파일] 메뉴의 [저장] 메뉴나 툴바에서 저장 아이콘()을 클릭하여 리파지토리 루트 폴더에 'getUSLServerParam.ozr' 파일로 저장합니다. 본 매뉴얼에서 리파지토리 루트 폴더는 'D:\Repository'입니다.

- 아래와 같은 html을 만들어 실행합니다.

```
<HTML>
<BODY>
<OBJECT    id    =    "ozviewer"    CLASSID="CLSID:0DEF32F8-170F-46f8-B1FF-
4BF7443F5F25" width="100%" height="100%">

<param name="connection.servlet" value="http://127.0.0.1:8080/oz35/server">
<param name="connection.reportname" value="/getUSLServerParam.ozr">
<param name="viewer.isframe" value="true">
<param name="information.debug" value="true">

</OBJECT>
</body>
</HTML>
```

- 실행된 보고서에서 라벨의 caption 값으로 설정한 USL패러미터 값이 표시되는지 확인합니다.





USL 예제 2 - PKI 모듈과의 연동 예제

USL을 이용하여 사용자의 웹 환경에 적용된 PKI 모듈과의 연동을 구현할 수 있습니다.

■ **USL 서버 구현**

사용자 웹 환경에 적용된 PKI 모듈에서 제공하는 클래스를 이용합니다.

아래 예는 오즈 서버에서 오즈 뷰어로 데이터를 보낼 때 암호화를 수행합니다.

서버에서 클라이언트로 암호화된 데이터를 전송하기 위해 Output Stream을 상속받는 클래스에서 write(), flush() 메소드를 재구현하였고, 데이터를 클라이언트로 전송할 때 호출되는 flush() 메소드에서 데이터를 암호화하였습니다.

오즈 서버는 데이터를 바이트 정보로 전송하기 때문에 암호화된 데이터를 다시 바이트 정보로 변환해야 합니다.

세션키 값은 암호화키로 사용되며, 암호화 부분을 처리하기 위한 함수는 PKI 모듈에 따라 다릅니다.

```
package ozusl;

import java.io.*;
import oz.cp.OZUSLServer;
import javax.servlet.http.HttpSession;

import java.security.*;
import java.security.spec.*;

import oz.cp.message.OzcmException;
import pki.lib.*;       // PKI 모듈 클래스 import

public class PKIUSLServer extends OZUSLServer
{
  protected String share_key_tag = null;
  protected String share_key_value = null;

  protected Secure _secu = null;
  protected EnvelopedMessage _se = null;

  public PKIUSLServer() {
  }

  class PKIUSLServerOutputStream extends OutputStream
  {
    ByteArrayOutputStream bout;
    DataOutputStream out_org = null;
    int b_length = 0;
```

```
public PKIUSLServerOutputStream(HttpSession _http_session,
                                String _report_name,
DataOutputStream _out_org) throws OzcmException
{

//////////////////////// Session 체크 ////////////////////////
share_key_value = (String)_http_session.getValue(share_key_tag);
  if (share_key_value==null)
    throw new OzcmException("session was not found");
    ////////////////////////////////////////////
    bout = new ByteArrayOutputStream();
    out_org = _out_org;
  }

  public void write(int b) throws IOException {
    bout.write(b);
    b_length++;
  }

  public void close() throws IOException {
  }

  public void flush() throws IOException {
    try{
   //////////////////////// 암호화 부분 ////////////////////////
      _secu = Secure.getInstance();
      _se = new EnvelopedMessage( _secu);
      String dummy=_se.encryptInit(secu.WEB, secu.KM, share_key_value);

      bout.flush();
     byte[] b_org = bout.toByteArray();

      byte[] encrypt_base64 = _se.encryptUpdate(b_org).getBytes();
      ////////////////////////////////////////
      out_org.writeInt(encrypt_base64.length);
      out_org.write(encrypt_base64, 0, encrypt_base64.length);
      out_org.flush();
    } catch(Exception e){
       e.printStackTrace();
       throw new IOException("PKI: error... " + e.toString());
    }
  }
}

public InputStream createSecureInputStream(DataInputStream org_in) throws
  OzcmException {
  try{
```

```
        share_key_tag = org_in.readUTF();
        if(share_key_tag.equals("param is null") || share_key_tag.equals("")
           || share_key_tag==null)
          share_key_tag="SESSION_ID";
        } catch(Exception e){
          share_key_tag = null;
          e.printStackTrace();
          throw new OzcmException(
              "PKI: can't get share_key ! check USL configuration for " +
              report_name);
        }
        return null;
    }

    public OutputStream createSecureOutputStream(DataOutputStream out_org)
    throws OzcmException {
        return new AAUSLServerOutputStream(http_session, report_name, out_org);
    }
}
```

■ **USL 클라이언트 구현**

USL 서버 모듈과 마찬가지로 PKI 모듈에서 제공하는 COM라이브러리를 이용합니다.

아래 예의 경우 서버에서 받은 데이터를 복호화하기 위해 CJDataInputStream 클래스의 생성자에서 스트림으로부터 모든 데이터를 읽어 복호화하였습니다.

그리고 각각의 메소드가 호출될 때마다 복호화된 데이터를 읽을 수 있도록 구현되었습니다.

```
<<USLClient.cpp>>
// USLClient.cpp: implementation of the CUSLClient class.
//
//////////////////////////////////////////////////////////////////////

#include <stdafx.h>
#include <ozuslclient/ozuslclient.h>
#include <ozuslclient/USLClient.h>
#include <ozuslclient/axmofe.h>
#include <ozuslclient/JUSLDataInputStream.h>
#include <ozuslclient/jusldataoutputstream.h>

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif

//////////////////////////////////////////////////////////////////////
// Construction/Destruction
```

```
//////////////////////////////////////////////////////////////////////

CUSLClient::CUSLClient()
{
  m_str_ip = _T("");
  m_str_url = _T("");
  m_str_pub_key_path = _T("");
  m_parameters = NULL;
}

CUSLClient::~CUSLClient()
{
}

void CUSLClient::SetServerIP(CString str_ip)
{
  m_str_ip = str_ip;
}
void CUSLClient::SetServerURL(CString str_url)
{
  m_str_url = str_url;
}

void CUSLClient::SetParameter(Parameter * parameters)
{
  m_parameters = parameters;
}

CJDataInputStream * CUSLClient::GetSecuredInputStream(CJDataInputStream *pIn)
{
  return new CJUSLDataInputStream(pIn,TRUE, m_str_pub_key_path);
}

CJDataOutputStream * CUSLClient::GetSecuredOutputStream(
CJDataOutputStream *pOut)
{
  CJUSLDataOutputStream out(pOut,FALSE, m_parameters);
  out.WriteSessionKeyToServer();
  m_str_pub_key_path  = out.GetPublicKeyPath();
  return pOut;
}

<<JUSLDataInputStream.cpp>>
// JUSLDataInputStream.cpp: implementation of the CJUSLDataInputStream class.
//
//////////////////////////////////////////////////////////////////////

#include "stdafx.h"
```

```
#include "JUSLDataInputStream.h"
#include <ozuslclient/axmofe.h>
#include <ozuslclient/base64.h>


#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif _DEBUG

//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////

CJUSLDataInputStream::CJUSLDataInputStream(CJDataInputStream *pIn,
BOOL _isShouldDelete,
CString str_pub_key) : CJDataInputStream(pIn, _isShouldDelete)
{
  try{
    Jint jsize = pIn->readInt();
    //byte* b_org = new byte[size];

    CJArray<Jbyte>* arrayBuffer;
    char * buf = new char[2048];
    int readSize = -1;
    CJByteArrayOutputStream bout;
    try{
      while(true) {
        readSize = pIn->read(buf,0,2048);
        if (readSize ==0) break;
        bout.write(buf,0,readSize);
      }
      arrayBuffer = bout.toByteArrayForSign();
      in->close();
      delete buf;
    }catch(CZException *eof){
    throw eof;
  }
  IAxMOFE_NCA pki;
  if(!pki.CreateDispatch(_T("AxMOFE.AxMOFE_NCA.1"), NULL))
  {
  }
  char * tmp = arrayBuffer->getBuffer();
  int ssize = arrayBuffer->length();

  CString decrypt_base64 =
          pki.SymmDecryptDataEx(tmp, pki.ReadDataFile(str_pub_key), _T("SEED"));
```

```
  byte* tmpByte2 = new byte[decrypt_base64.GetLength()];
  int dsie = decrypt_base64.GetLength();

  int size = 0;

  CTP_Base64Decode((unsigned char*)
    decrypt_base64.GetBuffer(decrypt_base64.GetLength()), tmpByte2, &size);

  m_pIn =
     new CJDataInputStream(new CJByteArrayInputStream((char*)tmpByte2,0,size,
                           TRUE), TRUE);
  in = m_pIn;

  delete arrayBuffer;
  }catch(CZException * ex){
    throw ex;
  }
}

CJUSLDataInputStream::~CJUSLDataInputStream()
{
/*
  if(m_pIn)
  {
    m_pIn->close();
    delete m_pIn;
  }
*/
}

void CJUSLDataInputStream::GetBytes (const CString str, byte* b, int len)
{
  for (int i = 0 ; i < len ; i++) {
    b[i] =str.GetAt (i);
  }
}

Jint CJUSLDataInputStream::available()
{
  return m_pIn->available();
}

Jbyte CJUSLDataInputStream::readByte()//throw (CJIOException *)
{
  return m_pIn->readByte();
}
```

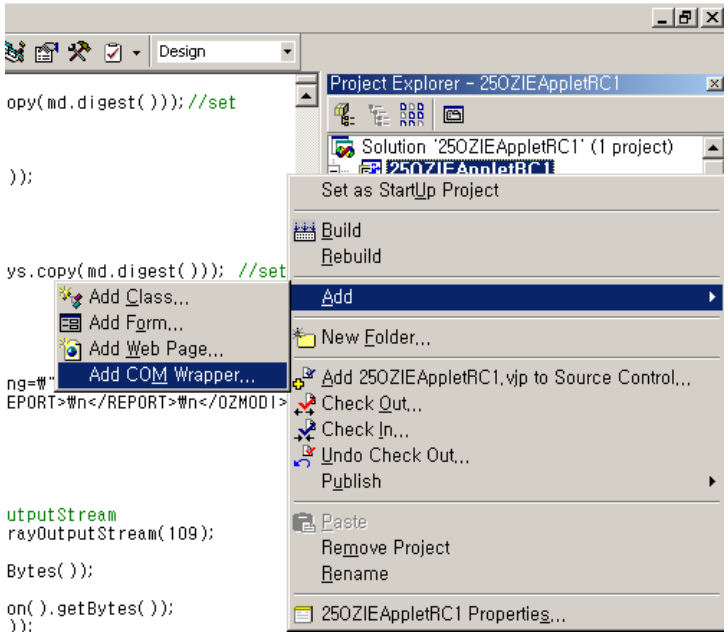
```
Jlong CJUSLDataInputStream::readLong()//throw (CJIOException *)
{
  return m_pIn->readLong();
}

Jdouble CJUSLDataInputStream::readDouble()//throw (CJIOException *)
{
  return m_pIn->readDouble();
}

Jfloat CJUSLDataInputStream::readFloat()//throw (CJIOException *)
{
  return m_pIn->readFloat();
}

Jboolean CJUSLDataInputStream::readBoolean()//throw (CJIOException *)
{
  return m_pIn->readBoolean();
}

Jshort CJUSLDataInputStream::readShort()//throw (CJIOException *)
{
  return m_pIn->readShort();
}

Jchar CJUSLDataInputStream::readChar()//throw (CJIOException *)
{
  return m_pIn->readChar();
}

Jint CJUSLDataInputStream::readInt()//throw (CJIOException *)
{
  return m_pIn->readInt();
}

CString CJUSLDataInputStream::readUTF()//throw (CJIOException *)
{
  return m_pIn->readUTF();
}

CString CJUSLDataInputStream::readString()//throw (CJIOException *)
{
  return m_pIn->readString();
}

Jint CJUSLDataInputStream::read(CJArray<Jbyte> &b)//throw (CJIOException *)
{
  return m_pIn->read(b);
```

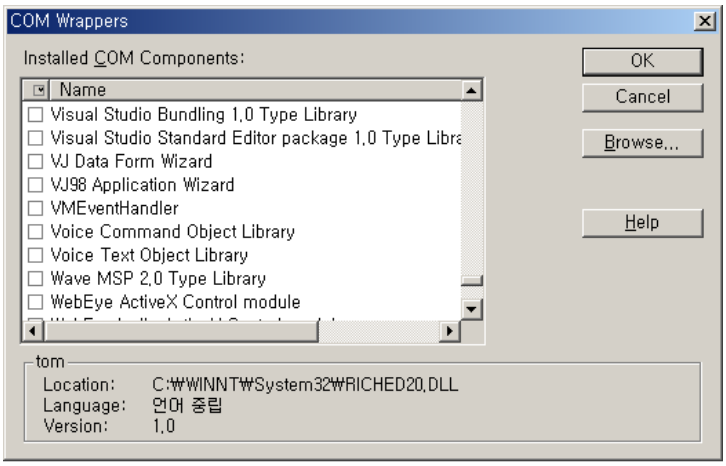
```
}

Jint CJUSLDataInputStream::read(CJArray<Jbyte> &b, Jint off, Jint len)
//throw (CJIOException *)
{
  return m_pIn->read(b, off, len);
}

int CJUSLDataInputStream::getRemainSize()
{
  return m_pIn->getRemainSize();
}

Jbyte * CJUSLDataInputStream::getBufferPointer()
{
  return m_pIn->getBufferPointer();
}

<<JUSLDataOutputStream.cpp>>
// JUSLDataOutputStream.cpp: implementation of the CJUSLDataOutputStream class.
//////////////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "JUSLDataOutputStream.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif _DEBUG

//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////
CJUSLDataOutputStream::CJUSLDataOutputStream(CJDataOutputStream *pOut,
BOOL _isShouldDelete,
Parameter *parameter) : CJDataOutputStream(pOut, _isShouldDelete)
{
  m_pOut = pOut;
  m_parameter = parameter;
  m_str_pub_key_path = _T("");
}

CJUSLDataOutputStream::~CJUSLDataOutputStream()
{
}

void CJUSLDataOutputStream::WriteSessionKeyToServer()
```

```
{
  try{
    BOOL b = TRUE;
    for (int i=0; i<m_parameter->GetSize(); i++) {
      // Read session key from viewer's parameter to wriete this information to
    // 오즈 서버
      if(m_parameter->ElementAt(i).name.CompareNoCase(_T("session_name")) == 0){
        m_pOut->writeUTF(m_parameter->ElementAt(i).value);
        b=FALSE;
      }
      //Read ksign's key path to decode the ecoded data from 오즈 서버
        if (m_parameter->ElementAt(i).name.CompareNoCase(_T("ksign_path")) == 0)
       {
         m_str_pub_key_path = m_parameter->ElementAt(i).value;
       }
     }
   }catch(CZException * e){
     throw e;
   }
}

CString CJUSLDataOutputStream::GetPublicKeyPath()
{
  return m_str_pub_key_path;
}
}
```

※ PKI 솔루션 관련 참고사항

- 일반적으로 PKI 솔루션의 클라이언트는 ActiveX 형태로 제공되는 경우가 많습니다. 이 경우 PKI 솔루션의 ActiveX 모듈에서 암•복호화할 때 byte[]형을 지원하는지 확인해야 합니다. 일반 웹 페이지에서는 사용자 아이디나 패스워드 같은 String 정보를 암•복호화하기 때문에 대부분 PKI 솔루션에서는 String 형태의 라이브러리를 제공합니다. 그러나 오즈는 데이터를 byte[]형태로 전송하므로 byte[]형을 지원하는 라이브러리를 제공받아야 합니다. 만약 String 형태로 데이터를 전달 받는다면 데이터를 base64로 인코딩해서 모듈을 제공받아야 합니다. 이는 데이터가 깨지는 것을 방지하기 위해서 입니다.
- Client PKI ActiveX 모듈 사용방법
  ① 클라이언트 PC에 해당 ActiveX 모듈을 설치합니다.
  ② VJ++의 COM Wrapper 클래스를 이용하여 PKI 모듈 컴포넌트를 등록합니다.





③ PKI 모듈 클래스를 Import하여 USL 클라이언트 모듈을 작성합니다.

Ⅵ. 사용자 컴포넌트

- 사용자 컴포넌트 개요
- 연동 방법
- 자바 인터페이스
- 사용자 컴포넌트 C 구현 함수 목록

본 장에서는 오즈 리포트에서 사용자 컴포넌트를 추가/개발하는 방법에 대해 설명합니다.

사용자 컴포넌트 개요

사용자 컴포넌트는 라벨의 그리기 타입 중의 하나입니다.
라벨의 그리기 타입을 사용자 컴포넌트로 하면 해당 모듈을 로딩하여 보고서 디자인시 사용하며, 실제 서비스할 때는 오즈 뷰어에서 사용합니다.

본 매뉴얼에서는 사용자 컴포넌트의 이름을 "PDF417"라고 가정하고 설명합니다.

연동 방법

사용자 컴포넌트 DLL 명명 규칙

명명규칙 : ozc_컴포넌트명.dll
만일 사용자 컴포넌트 이름을 " PDF 417"로 한다면 "ozc_pdf417.dll"로 명명하여야 합니다.

오즈 리포트 디자이너와의 연동

사용자 컴포넌트를 launch.cfg의 classpath에 추가한 후 사용합니다.

뷰어와의 연동

뷰어는 사용자 컴포넌트를 사용할 경우(화면 보기, 인쇄, 속성 편집 등)에 ozc_pdf417.dll를 WIN32 API의 LoadLibrary를 사용하여 로드해서 합니다.

DLL을 클라이언트에 배포시에는 ozuser.zip과 같이 dll을 zip으로 압축한 후 웹서버의 오즈 뷰어 탑재 위치(예: ozrviewer.idf 파일이 있는 위치)로 옮긴 후 ZTransfer에서 사용하는 설치 정보 파일(*.idf)에서 다운로드하여 설치합니다. 자세한 내용은 "오즈 리포트 뷰어 매뉴얼의 Ⅱ. 오즈 뷰어 설치 및 제거"를 참조하시기 바랍니다.

※ 주의사항 : 익스포트시 PDF와 SVG는 사용자 컴포넌트가 배경과 테두리만 저장되고 내용은 저장되지 않습니다. Excel, Word, Powerpoint, HTML 익스포트의 경우에는 이미지 형태로 저장됩니다.

자바 인터페이스

구현이 필요한 인터페이스

■ **interface oz.ucomp.OZUserComp**

- 설명 : 사용자 컴포넌트를 화면상에 그리는 클래스입니다.
- Method
  - public void setMeta(OZUserCompMeta meta)
    속성을 나타내는 OZUserCompMete 클래스를 등록하는 함수입니다.
  - public void paint(oz.client.IGraphics g, oz.client.OZPoint basePoint, oz.client.shape.ILabel label, String data)
    사용자 컴포넌트를 화면에 출력하는 함수입니다. Data 변수는 표시할 Caption 값입니다. 이 함수는 아무런 구현을 하지 않을 경우 오즈 리포트 디자이너의 해당 컴포넌트에서는 빈 라벨로 보입니다.
    ※ 참고사항 : Tiff 이미지 저장 시 이미지의 저장 위치를 설정하여 저장할 경우에는 paintEX 함수를 사용하시기 바랍니다.
  - public oz.client.OZDimension getAutosize(oz.client.shape.ILabel label, String data)
    라벨 속성의 자동 크기 조정이 '예'일 때 사용하는 함수입니다. 오즈 리포트 디자이너에서는 사용하지 않으므로 구현할 필요가 없지만, 컴파일을 위해서 원형으로는 추가되어야 합니다.

■ **interface oz.ucomp.OZUserCompMeta**

- 설명 : 사용자 컴포넌트의 속성을 관리하는 클래스입니다.
- Method
  - public void xmlBind(java.util.Hashtable t)
    오즈 리포트 디자이너에서 보고서 폼 저장시 속성을 저장하는 함수입니다.
  - public void setAttr(String s, String v)
    사용자 컴포넌트의 속성을 설정하는 함수입니다. (s : 속성명, v : 속성값)
  - public void writeMe(oz.framework.cp.io.OZDataOutputStream out)
    사용자 컴포넌트의 속성을 직렬화(Serialize)해서 out 스트림으로 쓰는 함수입니다. 오즈 리포트 디자이너에서는 사용하지 않으므로 구현할 필요가 없지만, 컴파일을 위해서 원형으로는 추가되어야 합니다.

- public void readMe(oz.framework.cp.io.OZDataInputStream in)
  사용자 컴포넌트의 속성을 in스트림에서 읽어서 직렬화(Serialize)하는 함수입니다. 오즈 리포트 디자이너에서는 사용하지 않으므로 구현할 필요가 없지만, 컴파일을 위해서 원형으로는 추가되어야 합니다.

■ **interface oz.ucomp.OZUserCompMetaBeanInfo**

- 설명 : 사용자 컴포넌트의 속성을 디자이너로 알려주는 클래스입니다.
- Method
  - public PropertyDescriptor[] getPropertys()
    오즈 리포트 디자이너의 속성 창에서 표시할 수 있도록 PropertyDescriptor들을 반환합니다.

관련 Class

■ **oz.client.OZPoint, oz.client.OZDimension**
java.awt.Point와 java.awt.Dimension의 float형 버전

■ **oz.framework.cp.io.OZDataOutputStream,**
oz.framework.cp.io.OZDataInputStream
java.io.DataOutputStream와 java.io.OZDataInputStream와 동일함

■ **oz.client.shape.ILabel**
라벨 객체를 나타내는 인터페이스

■ **oz.client.IGraphics**
java.awt.Graphics를 Wrapping한 클래스

예제

PDF417은 2D 바코드의 한 형태입니다. 다음은 바코드 이미지를 생성하여 출력하는 예제들입니다.

■ **PDF417.java**

```
package ucomp;
import oz.ucomp.*;

public class PDF417 implements OZUserComp{
   protected PDF417Meta meta = null;
   public void setMeta(OZUserCompMeta meta){
     // PDF417Meta settting
     this.meta = (PDF417Meta)meta;
   }
   public void paint(oz.client.IGraphics g, oz.client.OZPoint basePoint,
   oz.client.shape.ILabel label, String data)throws OZUserCompException{
     // fill black rectangle
     g.setOZColor(oz.client.OZColor.black);
     g.fillRect(basePoint.x, basePoint.y, label.getWidth(), label.getHeight());
   }
   public oz.client.OZDimension getAutosize(oz.client.shape.ILabel label,
   String data) throws OZUserCompException{
     // no effect
     return new oz.client.OZDimension(label.getWidth(), label.getHeight());
   }
}
```

■ **PDF417Meta.java**

```
package ucomp;

import oz.ucomp.*;

public class PDF417Meta extends OZUserCompMeta{
   //propertyNames is text for designer property name
   public final static String[] propertyNames =
    new String[]{"행수","열수","에러교정","X크기","Y크기", "dll 이름", "dll URL",
    "dll파일크기"};
   public PDF417Meta() {
   }
   public void setAttr(String s, String v)throws OZUserCompException{
     //attribute setting function
     // s is attribute name
     // v is attribute value string
     if(s.equalsIgnoreCase("PDF417_ROWNUM")){
       setRowNum( Integer.parseInt(v) );
     }else if(s.equalsIgnoreCase("PDF417_COLNUM")){
       setColNum( Integer.parseInt(v) );
     }else if(s.equalsIgnoreCase("PDF417_ECC")){
       setECC( Integer.parseInt(v) );
     }else if(s.equalsIgnoreCase("PDF417_XSCALE")){
       setXScale( Integer.parseInt(v) );
```

```
  }else if(s.equalsIgnoreCase("PDF417_YSCALE")){
    setYScale( Integer.parseInt(v) );
  }else if(s.equalsIgnoreCase("PDF417_DLLNAME")){
    setDllName( v );
  }else if(s.equalsIgnoreCase("PDF417_DLLURL")){
    setDllURL( v );
  }else if(s.equalsIgnoreCase("PDF417_DLLSIZE")){
    setDllSize( Integer.parseInt(v) );
  }
}
public void xmlBind(java.util.Hashtable t)throws OZUserCompException{
  // attribute save function
  // Hashtable key is attribute name
  // Hashtable value is attribute value string
    t.put("PDF417_ROWNUM", String.valueOf(getRowNum()));
    t.put("PDF417_COLNUM", String.valueOf(getColNum()));
    t.put("PDF417_ECC", String.valueOf(getECC()));
    t.put("PDF417_XSCALE", String.valueOf(getXScale()));
    t.put("PDF417_YSCALE", String.valueOf(getYScale()));
    t.put("PDF417_DLLNAME", getDllName());
    t.put("PDF417_DLLURL", getDllURL());
    t.put("PDF417_DLLSIZE", String.valueOf(getDllSize()));
}
public void writeMe(oz.framework.cp.io.OZDataOutputStream out) throws
Exception{
  //serialize write function
  out.writeInt(rowNum);
  out.writeInt(colNum);
  out.writeInt(ecc);
  out.writeInt(xScale);
  out.writeInt(yScale);
  out.writeInt(dllSize);
  out.writeUTF2(dllName);
  out.writeUTF2(dllURL);
}
public void readMe(oz.framework.cp.io.OZDataInputStream in)throws Exception{
   //serialize read function
   rowNum = in.readInt();
   colNum = in.readInt();
   ecc = in.readInt();
   xScale = in.readInt();
   yScale = in.readInt();
   dllSize = in.readInt();
   dllName = in.readUTF2();
   dllURL = in.readUTF2();
}
int rowNum = 0;
int colNum = 8;
```

```
int ecc = 0;
int xScale = 1;
int yScale = 1;
String dllName = "";
String dllURL = "";
int dllSize = 67000;
public int getRowNum(){
    return rowNum;
}
public int getColNum(){
    return colNum;
}
public int getECC(){
    return ecc;
}
public int getXScale(){
    return xScale;
}
public int getYScale(){
    return yScale;
}
public String getDllName(){
    return dllName;
}
public String getDllURL(){
    return dllURL;
}
public int getDllSize(){
    return dllSize;
}
//firePropertyChangListener is fuction for attribute change event
public void setRowNum(int i){
    if(i < 0)return;
    int oldValue = getRowNum();
    rowNum = i;
    firePropertyChangListener(propertyNames[0], new Integer(oldValue),
                              new Integer(i));
}
public void setColNum(int i){
    if(i < 0)return;
    int oldValue = getColNum();
    colNum = i;
    firePropertyChangListener(propertyNames[1], new Integer(oldValue),
                              new Integer(i));
}
public void setECC(int i){
    if(i < 0)return;
    int oldValue = getECC();
```

```
        ecc = i;
        firePropertyChangListener(propertyNames[2], new Integer(oldValue),
                                  new Integer(i));
    }
    public void setXScale(int i){
        if(i < 1)return;
        int oldValue = getXScale();
        xScale = i;
        firePropertyChangListener(propertyNames[3], new Integer(oldValue),
                                  new Integer(i));
    }
    public void setYScale(int i){
        if(i < 1)return;
        int oldValue = getYScale();
        yScale = i;
        firePropertyChangListener(propertyNames[4], new Integer(oldValue),
                                  new Integer(i));
    }
    public void setDllName(String i){
        String oldValue = getDllName();
        dllName = i;
        firePropertyChangListener(propertyNames[5], oldValue, i);
    }
    public void setDllURL(String i){
        String oldValue = getDllURL();
        dllURL = i;
        firePropertyChangListener(propertyNames[6], oldValue, i);
    }
    public void setDllSize(int i){
        if(i < 1)return;
        int oldValue = getDllSize();
        dllSize = i;
        firePropertyChangListener(propertyNames[7], new Integer(oldValue),
                                  new Integer(i));
    }
}
```

- **PDF417MetaBeanInfo.java**

```
package ucomp;

import java.beans.*;
import oz.ucomp.OZUserCompMetaBeanInfo;

public class PDF417MetaBeanInfo extends OZUserCompMetaBeanInfo{

    private static Class beanClass = PDF417Meta.class;
    private static PropertyDescriptor[] rv = null;
```

```
   private static String error = null;
   public PropertyDescriptor[] getPropertys() {
      if(rv == null) throw new Error(beanClass.getName() + " : " + error);
      return rv;
   }
   static{
     try
      {
////////////////////////////////////////////////////////////////////////
//PDF417Meta.propertyNames[0] : Attribute Text
//"getRowNum": Attribute get fuction in PDF417Meta
//"setRowNum": Attribute set fuction in PDF417Meta

//setShortDescription have to start "cb"
////////////////////////////////////////////////////////////////////////

   PropertyDescriptor rowNum = new PropertyDescriptor
   (PDF417Meta.propertyNames[0], beanClass, "getRowNum", "setRowNum");
   rowNum.setShortDescription("cba");
   rowNum.setConstrained(true);

   PropertyDescriptor colNum = new PropertyDescriptor
   (PDF417Meta.propertyNames[1], beanClass, "getColNum", "setColNum");
   colNum.setShortDescription("cbb");
   colNum.setConstrained(true);

   PropertyDescriptor ecc = new PropertyDescriptor
   (PDF417Meta.propertyNames[2], beanClass, "getECC", "setECC");
   ecc.setShortDescription("cbc");
   ecc.setConstrained(true);

   PropertyDescriptor xscale = new PropertyDescriptor
   (PDF417Meta.propertyNames[3], beanClass, "getXScale", "setXScale");
   xscale.setShortDescription("cbd");
   xscale.setConstrained(true);

   PropertyDescriptor yscale = new PropertyDescriptor
   (PDF417Meta.propertyNames[4], beanClass, "getYScale", "setYScale");
   yscale.setShortDescription("cbe");
   yscale.setConstrained(true);

   PropertyDescriptor dllName = new PropertyDescriptor
   (PDF417Meta.propertyNames[5], beanClass, "getDllName", "setDllName");
   dllName.setShortDescription("cbf");
   dllName.setConstrained(true);

   PropertyDescriptor dllUrl = new PropertyDescriptor
   (PDF417Meta.propertyNames[6], beanClass, "getDllURL", "setDllURL");
```

```
    dllUrl.setShortDescription("cbg");
    dllUrl.setConstrained(true);

    PropertyDescriptor dllSize = new PropertyDescriptor
   (PDF417Meta.propertyNames[7], beanClass, "getDllSize", "setDllSize");
    dllSize.setShortDescription("cbh");
    dllSize.setConstrained(true);

    rv = new PropertyDescriptor[]{
    rowNum, colNum, ecc, xscale, yscale, dllName, dllUrl, dllSize};
    }catch(IntrospectionException e){
        rv = null;
        error = e.toString();
    }
  }
}
```

사용자 컴포넌트 C 구현 함수 목록

오즈 뷰어에서 사용자 컴포넌트를 사용하기 위해서는 다음의 함수들을 구현해야 합니다.

구현 함수

■ **생성, 복사, 삭제 함수**

- GetNewInstance

Prototype	`DWORD GetNewInstance()`
Definition	사용자 컴포넌트 객체 생성 함수입니다.
Argument	없음
Return	사용자 컴포넌트 객체의 핸들을 반환, 에러 발생시 0(NULL)을 반환

- GetCopyInstance

Prototype	`DWORD GetCopyInstance(DWORD src)`
Definition	사용자 컴포넌트 객체 복사 함수입니다.
Argument	*src* 사용자 컴포넌트 객체 핸들
Return	복사된 객체의 핸들을 반환, 에러 발생시 0(NULL)을 반환

- DeleteInstance

Prototype	`void DeleteInstance(DWORD src)`
Definition	사용자 컴포넌트 객체 삭제 함수입니다.
Argument	*src* 사용자 컴포넌트 객체 핸들
Return	없음

■ **속성 Serialize 함수**

- getAttrListLength

Prototype	`int getAttrListLength(DWORD src)`
Definition	사용자 컴포넌트가 사용하는 속성들 개수를 반환하는 함수입니다.
Argument	*src* 사용자 컴포넌트 객체 핸들

Return	속성 개수를 반환 반환, 에러 발생시 -1을 반환

- getAttrList

Prototype	`BOOL getAttrList(DWORD src, char** attrs, const int length)`	
Definition	사용자 컴포넌트에서 사용하는 속성명들을 배열에 넣어서 attrs로 돌려줍니다.	
Argument	*src*	사용자 컴포넌트 객체 핸들
	attrs	속성명이 반환되는 버퍼의 Pointer
	length	getAttrListLength를 호출하여 나온 속성 개수
Return	성공하면 true를 반환, 에러 발생시 false를 반환	

- getAttrLength

Prototype	`int getAttrLength(DWORD src, const char* name)`	
Definition	입력된 속성값의 길이를 반환하는 함수입니다.	
Argument	*src*	사용자 컴포넌트 객체 핸들
	name	속성명
Return	속성값의 길이를 반환, 에러 발생시(자신의 속성이 아닌 경우) -1을 반환	

- getAttr

Prototype	`BOOL getAttr(DWORD src, const char* name, char* value, const int value_length)`	
Definition	name에 해당하는 속성의 값을 value로 반환하는 함수입니다.	
Argument	*src*	사용자 컴포넌트 객체 핸들
	name	속성명
	value	속성의 값을 저장하는 장소
	value_length	getAttrLength를 호출하여 나온 값
Return	성공하면 true를 반환, 에러 발생시(자신의 속성이 아닌 경우) false를 반환	

- setAttr

Prototype	`BOOL setAttr(DWORD src, const char* name, const char* value)`	
Definition	입력된 속성명의 속성값을 변경하는 함수입니다.	
Argument	*src*	사용자 컴포넌트 객체 핸들

	name	속성명
	value	변경하려는 속성의 값
Return	속성값의 길이를 반환, 에러 발생시(자신의 속성이 아닌 경우) -1을 반환	

- writeMe

Prototype	`char* writeMe(DWORD src, int * length)`	
Definition	사용자 컴포넌트의 속성을 serialize하는 함수입니다.	
Argument	*src*	사용자 컴포넌트 객체 핸들
	length	serialize된 binary data의 길이를 반환하기 위한 int형 변수의 pointer
Return	serialize된 binary data의 pointer를 반환, 에러 발생시 NULL을 반환	

- readMe

Prototype	`void readMe(DWORD src, const char* pData, const int length)`	
Definition	사용자 컴포넌트의 속성을 serialize된 binary data로부터 읽는 함수입니다.	
Argument	*src*	사용자 컴포넌트 객체 핸들
	pData	속성값을 읽어들이기 위한 binary data 저장소
	length	읽어들일 데이터의 길이
Return	없음	

■ **기타 함수**

- paint

Prototype	`void paint(DWORD src, HDC hDC, LPCTSTR data, const float x, const float y, const float w, const float h, const float scale)`	
Definition	화면에 그릴 때 호출되는 함수입니다.	
Argument	*src*	사용자 컴포넌트 객체 핸들
	hDC	그려질 화면의 핸들(WIN32 API 참조)
	data	라벨의 Caption
	x	컴포넌트의 x 좌표
	y	컴포넌트의 y 좌표
	w	컴포넌트의 너비
	h	컴포넌트의 높이

	scale	확대 값
Return	없음	

- paintEX

Prototype	`void paintEX(__cdecl *paintEX)(DWORD src, HDC hDC, LPCTSTR data, const float x, const float y, const float w, const float h, const float scale const float x_offset, const float y_offset)`	
Definition	화면에 그릴 때 호출되는 함수로 실제 이미지가 저장될 offset 값을 전달할 수 있습니다.	
Argument	*src*	사용자 컴포넌트 객체 핸들
	hDC	그려질 화면의 핸들(WIN32 API 참조)
	data	라벨의 Caption
	x	컴포넌트의 x 좌표
	y	컴포넌트의 y 좌표
	w	컴포넌트의 너비
	h	컴포넌트의 높이
	scale	확대 값
	x_offset	실제 이미지가 저장될 x offset
	y_offset	실제 이미지가 저장될 y offset
Return	없음	

- print

Prototype	`void print(DWORD src, HDC hDC, LPCTSTR data, const float x, const float y, const float w, const float h, const float scale, const float x_offset, const float y_offset)`	
Definition	인쇄할 때 호출되는 함수입니다.	
Argument	*src*	사용자 컴포넌트 객체 핸들
	hDC	그려질 화면의 핸들(WIN32 API 참조)
	data	라벨의 Caption
	x	컴포넌트의 x 좌표
	y	컴포넌트의 y 좌표
	w	컴포넌트의 너비
	h	컴포넌트의 높이

	scale	확대 값
	x_offset	프린트 x Offset
	y_offset	프린트 x Offset
Return	없음	

- getAutosize

Prototype	`void getAutosize(DWORD src, HDC hDC, LPCTSTR data, float * w, float * h)`	
Definition	자동크기조정을 사용할 때 사용자 컴포넌트의 가로, 세로 크기를 설정하는 함수입니다.	
Argument	*src*	사용자 컴포넌트 객체 핸들
	hDC	그려질 화면의 핸들(WIN32 API 참조)
	data	라벨의 Caption
	w	컴포넌트의 원래 너비의 pointer
	h	컴포넌트의 원래 높이의 pointer
Return	w, h에 자동크기 조정된 가로와 세로 설정값을 반환, 에러 발생시 반환값 없음	

예제

다음의 예제들은 C++를 이용한 정규 dll로서 PDF 417 사용자 컴포넌트 예제들입니다.

■ PDF417.h

```
// hanja.h : main header file for the HANJA DLL
//

#if !defined(AFX_CPdf417App_H__7FDF11DA_5BF8_4C07_9BCF_BB1C08A763FA__INCLUDED_)
#define AFX_CPdf417App_H__7FDF11DA_5BF8_4C07_9BCF_BB1C08A763FA__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#ifndef __AFXWIN_H__
      #error include 'stdafx.h' before including this file for PCH
#endif
```

```
#include <resource.h>
// main symbols
// CPdf417App
// See CPdf417App.cpp for the implementation of this class
//

class CPdf417App : public CWinApp
{
public:
  CPdf417App();

// Overrides
// ClassWizard generated virtual function overrides
//{{AFX_VIRTUAL(CPdf417AppaApp)
//}}AFX_VIRTUAL

//{{AFX_MSG(CPdf417App)
// NOTE - the ClassWizard will add and remove member functions here.
//    DO NOT EDIT what you see in these blocks of generated code !
//}}AFX_MSG
  DECLARE_MESSAGE_MAP()
};


//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately
//before the previous line.

#endif
// !defined(AFX_CPdf417App_H__7FDF11DA_5BF8_4C07_9BCF_BB1C08A763FA__INCLUDED_)
```

■ **PDF417.cpp**

```
// OZPdf417Comp.cpp : Defines the initialization routines for the DLL.
//

#include <stdafx.h>
#include <pdf417.h>

////////////////////
//PDF417 Barcode api include
#include "pdfapi.h"
////////////////////
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
```

```
#endif
////////////////////////////////////////////////////////////////////////////
////////
//PDF417 Barcode api implement
static PDFObject g_MacroObjIn, g_MacroObjOut;

PDFSIZET Encode(LPCTSTR dataStr, unsigned short rowNum, unsigned short colNum,
unsigned short ecc, UINT xScale, UINT yScale)
{
  PDFSIZET nInputLen = 0;
  char szAspect[ 15 ];
  UINT nDPI = 300;
  PDFSetDefaults();
  PDFBinaryMode( false );
  PDFSetSymbolStyle( STYLE_NORMAL );
  PDFSetECCLevel( ecc );
  if(rowNum == 0 && colNum == 0){
    PDFSetAspect( "1:2" );
  }else{
    PDFSetDimType( USE_FIXED );
    PDFSetRowCol( rowNum, colNum );
  }
  sprintf( szAspect, "%d:%d", xScale+yScale-1, xScale );
  PDFSetRowHeight( szAspect );

  MPDFDisable();
  nInputLen = (PDFSIZET)lstrlen( dataStr );
  MakeMemoryObject( &g_MacroObjIn, PDFINPUT, (LPSTR)dataStr );
  uint16 wParam = 0;
  uint32 lParam = MakeLParam( nDPI, xScale );
  wParam |= DIB_ADDFILEHEADER;
/*
  if ( pSettings->nWShave )
    wParam |= ShaveWidthOption( pSettings->nWShave );
    if ( pSettings->nHShave )
      wParam |= ShaveHeightOption( pSettings->nHShave );
*/
     PDFOutputAsDIB( lParam, wParam );
     return nInputLen;
}
int EncodeData(LPCTSTR dataStr, BYTE * rt, unsigned short rowNum,
unsigned short colNum, unsigned short ecc, UINT xScale, UINT yScale)
{
PDFSIZET nInputLen = Encode(dataStr, rowNum, colNum, ecc, xScale, yScale);
   BOOL fStatus;
   PDFSIZET rtsize = -1;
   //MakeFilenameObject( &g_MacroObjOut, PDFOUTPUT,  fileStr);
   MakeMemoryObject( &g_MacroObjOut, PDFOUTPUT, (LPVOID) rt);
```

```
  if ( PDFEncode( &g_MacroObjIn, nInputLen ) >= 0 )  {
     fStatus = (PDFMakeImage( &g_MacroObjOut, &rtsize ) == 0);
  }else{
     fStatus = FALSE;
  }
  if(!fStatus){
    rtsize = -1;
    PDFAbort( &g_MacroObjIn );
    PDFAbort( &g_MacroObjOut);
  }
  //printf(dataStr);
  return rtsize;
}
LPPICTURE EncodePrint(LPCTSTR dataStr, unsigned short rowNum,
unsigned short colNum, unsigned short ecc, UINT xScale, UINT yScale)
{
PDFSIZET nInputLen = Encode(dataStr, rowNum, colNum, ecc, xScale, yScale);
  BOOL fStatus;
  PDFSIZET rtsize = -1;

  HGLOBAL hGlobal = NULL;
  DWORD dwFileSize = 1024*10;

  LPVOID pvData = NULL;
  // alloc memory based on file size
  hGlobal = GlobalAlloc(GMEM_MOVEABLE, dwFileSize);
  if(NULL == hGlobal){
    PDFAbort( &g_MacroObjIn );
    PDFAbort( &g_MacroObjOut);
    return NULL;
  }

  pvData = GlobalLock(hGlobal);
  if(NULL == pvData){
  PDFAbort( &g_MacroObjIn );
  PDFAbort( &g_MacroObjOut);
    return NULL;
  }
  MakeMemoryObject( &g_MacroObjOut, PDFOUTPUT, pvData);

  if ( PDFEncode( &g_MacroObjIn, nInputLen ) >= 0 )  {
     fStatus = (PDFMakeImage( &g_MacroObjOut, &rtsize ) == 0);
  }else{
     fStatus = FALSE;
  }
  if(!fStatus){
```

```
   PDFAbort( &g_MacroObjIn );
   PDFAbort( &g_MacroObjOut);
   GlobalUnlock(hGlobal);
   return NULL;
  }

  GlobalUnlock(hGlobal);
  LPSTREAM pstm = NULL;
  // create IStream* from global memory
  HRESULT hr = CreateStreamOnHGlobal(hGlobal, TRUE, &pstm);
  if(!SUCCEEDED(hr)  || pstm == NULL){
   PDFAbort( &g_MacroObjIn );
   PDFAbort( &g_MacroObjOut);
   return NULL;
  }

  // Create IPicture from image file
  LPPICTURE gpPicture = NULL;
  hr = ::OleLoadPicture(pstm, dwFileSize, FALSE, IID_IPicture,
                        (LPVOID *)&gpPicture);

  if(!SUCCEEDED(hr)  || gpPicture == NULL){
   BOOL b1 = hr == E_POINTER;
   BOOL b2 = hr == E_NOINTERFACE;
   BOOL b3 = hr == E_OUTOFMEMORY;
   BOOL b4 = hr == E_UNEXPECTED;
   pstm->Release();
   PDFAbort( &g_MacroObjIn );
   PDFAbort( &g_MacroObjOut);
   return NULL;
}
   pstm->Release();

  return gpPicture;
}
/////////////////////////////////////////////////////////////////////////////

//
//Note!
//
//If this DLL is dynamically linked against the MFC
//DLLs, any functions exported from this DLL which
//call into MFC must have the AFX_MANAGE_STATE macro
//added at the very beginning of the function.
//
//For example:
//
//extern "C" BOOL PASCAL EXPORT ExportedFunction()
```

```
//{
//AFX_MANAGE_STATE(AfxGetStaticModuleState());
// normal function body here
//}
//
//It is very important that this macro appear in each
//function, prior to any calls into MFC.  This means that
//it must appear as the first statement within the
//function, even before any object variable declarations
//as their constructors may generate calls into the MFC
//DLL.
//
//Please see MFC Technical Notes 33 and 58 for additional
//details.
//

/////////////////////////////////////////////////////////////////////////////
// CHanjaApp

BEGIN_MESSAGE_MAP(CPdf417App, CWinApp)
//{{AFX_MSG_MAP(CPdf417App)
// NOTE - the ClassWizard will add and remove mapping macros here.
// DO NOT EDIT what you see in these blocks of generated code!
//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CPdf417App construction

CPdf417App::CPdf417App()
{
// TODO: add construction code here,
// Place all significant initialization in InitInstance
}

/////////////////////////////////////////////////////////////////////////////
// The one and only CPdf417App object

CPdf417App theApp;


/////////////////////////////////////////////////////////////////////////////
// The one and only OZPdf417Comp object
class OZPdf417Comp{
public:
//////////////////////////////
//constructor
OZPdf417Comp();
```

```
////////////////////////////////
////////////////////////////////
//copy constructor
OZPdf417Comp(OZPdf417Comp & cp);
////////////////////////////////
////////////////////////////////
//destructor
virtual ~OZPdf417Comp();
////////////////////////////////

////////////////////////////////
//function for Attributes
int getAttrListLength();
BOOL getAttrList(TCHAR** attrs, const int length);
int getAttrLength(CString name);
BOOL getAttr(CString name, TCHAR* value, const int value_length);
BOOL setAttr(CString name, CString value);
char* writeMe(int * length);
void readMe(const char* pData, const int length);
////////////////////////////////
////////////////////////////////
//function for paint
void paint(HDC hDC, CString data, const float x, const float y, const float w,
const float h, const float scale);
////////////////////////////////
////////////////////////////////
//function for print
void print(HDC hDC, CString data, const float x, const float y, const float w,
const float h, const float scale, const float x_offset, const float y_offset);
////////////////////////////////
////////////////////////////////
//function for autosize
void getAutosize(HDC hDC, CString data, float * w, float * h);
////////////////////////////////
private:
////////////////////////////////
//Attributes
  int rowNum;
  int colNum;
  int ecc;
  int xScale;
  int yScale;
////////////////////////////////
};


//////////////////////////////////////////////////////////////////////////
// c api for oz viewer
```

```
#ifdef __cplusplus
extern "C"
{
#endif

__declspec( dllexport ) DWORD __cdecl GetNewInstance()
{
  return (DWORD)(void*)new OZPdf417Comp();
}
__declspec( dllexport ) DWORD __cdecl GetCopyInstance(DWORD src)
{
 return (DWORD)(void*)new OZPdf417Comp(*((OZPdf417Comp *)(void *)src));
}
__declspec( dllexport ) void __cdecl DeleteInstance(DWORD src)
{
  delete ((OZPdf417Comp *)(void *)src);
}
////////////////////////////////////////////////////////////////////////
__declspec( dllexport ) int __cdecl getAttrListLength(DWORD src)
{
  return ((OZPdf417Comp *)(void *)src)->getAttrListLength();
}
__declspec( dllexport ) BOOL __cdecl getAttrList(DWORD src, TCHAR** attrs,
const int length){
  return ((OZPdf417Comp *)(void *)src)->getAttrList(attrs, length);
}
__declspec( dllexport ) int __cdecl getAttrLength(DWORD src, const TCHAR* name)
{
  return ((OZPdf417Comp *)(void *)src)->getAttrLength(name);
}
__declspec( dllexport ) BOOL __cdecl getAttr(DWORD src, const TCHAR* name,
TCHAR* value, const int value_length){
  return ((OZPdf417Comp *)(void *)src)->getAttr(name, value, value_length);
}
__declspec( dllexport ) BOOL __cdecl setAttr(DWORD src, const TCHAR* name,
const TCHAR* value)
{
  return ((OZPdf417Comp *)(void *)src)->setAttr(name, value);
}
__declspec( dllexport ) char* __cdecl writeMe(DWORD src, int * length)
{
  return ((OZPdf417Comp *)(void *)src)->writeMe(length);
}
__declspec( dllexport ) void __cdecl readMe(DWORD src, const char* pData,
const int length)
{
  ((OZPdf417Comp *)(void *)src)->readMe(pData, length);
}
```

```
__declspec( dllexport ) void __cdecl paint(DWORD src, HDC hDC, LPCTSTR data,
const float x, const float y, const float w, const float h, const float scale)
{
  ((OZPdf417Comp *)(void *)src)->paint(hDC, data, x, y, w, h, scale);
}
__declspec( dllexport ) void __cdecl print(DWORD src, HDC hDC, LPCTSTR data,
const float x, const float y, const float w, const float h, const float scale,
const float x_offset, const float y_offset)
{
((OZPdf417Comp *)(void *)src)->print(hDC, data, x, y, w, h, scale, x_offset,
y_offset);
}
__declspec( dllexport ) void __cdecl getAutosize(DWORD src, HDC hDC,
LPCTSTR data, float * w, float * h)
{
  ((OZPdf417Comp *)(void *)src)->getAutosize(hDC, data, w, h);
}
#ifdef __cplusplus
}
#endif
////////////////////////////////////////////////////////////////////////////
// int to string convert function
CString _toString(int i){
  CString rst;
  rst.Format(_T("%d"), i);
  return rst;
}

////////////////////////////////////////////////////////////////////////////
// OZPdf417Comp implement

//////////////////////////////
//constructor
OZPdf417Comp::OZPdf417Comp(){
    rowNum = 0;
    colNum = 8;
    ecc = 0;
    xScale = 1;
    yScale = 1;
}
//////////////////////////////
//////////////////////////////
//copy constructor
OZPdf417Comp::OZPdf417Comp(OZPdf417Comp & comp){
    rowNum = comp.rowNum;
    colNum = comp.colNum;
    ecc = comp.ecc;
    xScale = comp.xScale;
```

```
   yScale = comp.yScale;
}
////////////////////////////
////////////////////////////
//destructor
OZPdf417Comp::~OZPdf417Comp(){
}
////////////////////////////

int OZPdf417Comp::getAttrListLength(){
// Attribute count return
  return 5;
}
BOOL OZPdf417Comp::getAttrList(TCHAR** attrs, const int length){
// Attribute names set to attrs
// length is result of getAttrListLength() function call
  if(length < 0)return FALSE;
  int nIndex = 0;
  if(length == nIndex)return TRUE;
  attrs[nIndex++] = _T("PDF417_ROWNUM");
  if(length == nIndex)return TRUE;
  attrs[nIndex++] = _T("PDF417_COLNUM");
  if(length == nIndex)return TRUE;
  attrs[nIndex++] = _T("PDF417_ECC");
  if(length == nIndex)return TRUE;
  attrs[nIndex++] = _T("PDF417_XSCALE");
  if(length == nIndex)return TRUE;
  attrs[nIndex++] = _T("PDF417_YSCALE");
  return TRUE;
}

int OZPdf417Comp::getAttrLength(CString name){
  // name is Attribute name
  // return TCHAR length of value by name
  CString value;
  if(name == "PDF417_ROWNUM"){
     value = _toString(rowNum);
  }else if(name == "PDF417_COLNUM"){
     value = _toString(colNum);
  }else if(name == "PDF417_ECC"){
     value = _toString(ecc);
  }else if(name == "PDF417_XSCALE"){
     value = _toString(xScale);
  }else if(name == "PDF417_YSCALE"){
     value = _toString(yScale);
////////////////////////////////////////////////////
  }else{
    return -1;
```

```
  }
    return value.GetLength()+1;
}

BOOL OZPdf417Comp::getAttr(CString name, TCHAR* valueBuffer,
const int value_length){
// name is Attribute name
// Attribute value set to valueBuffer
// value_length is result of getAttrLength() function call
    CString value;
    if(name == _T("PDF417_ROWNUM")){
        value = _toString(rowNum);
    }else if(name == _T("PDF417_COLNUM")){
        value = _toString(colNum);
    }else if(name == _T("PDF417_ECC")){
        value = _toString(ecc);
    }else if(name == _T("PDF417_XSCALE")){
        value = _toString(xScale);
    }else if(name == _T("PDF417_YSCALE")){
        value = _toString(yScale);
////////////////////////////////////////////////////
    }else{
    // return false if can`t find attribute
        return FALSE;
    }
    if(value.GetLength() >= value_length-1){
    // return false if length is differ
        return FALSE;
    }
    _tcscpy(valueBuffer, value);
    return TRUE;
}
BOOL OZPdf417Comp::setAttr(CString name, CString value){
// name is Attribute name
// value is Attribute value
    if(name == _T("PDF417_ROWNUM")){
        rowNum = _ttoi(value);
    }else if(name == _T("PDF417_COLNUM")){
        colNum = _ttoi(value);
    }else if(name == _T("PDF417_ECC")){
        ecc = _ttoi(value);
    }else if(name == _T("PDF417_XSCALE")){
        xScale = _ttoi(value);
    }else if(name == _T("PDF417_YSCALE")){
        yScale = _ttoi(value);
    }else{
    // return false if can`t find attribute
        return FALSE;
```

```
      return TRUE;
}
char* OZPdf417Comp::writeMe(int * length){
// serialize functin
// binary length set to length
// return my binary
*length = 20;
   char* pData = new char[*length];
   int i = 0;
   pData[i++] = (char)((rowNum & 0xFF000000) >> 24);
   pData[i++] = (char)((rowNum & 0x00FF0000) >> 16);
   pData[i++] = (char)((rowNum & 0x0000FF00) >> 8);
   pData[i++] = (char)((rowNum & 0x000000FF));

   pData[i++] = (char)((colNum & 0xFF000000) >> 24);
   pData[i++] = (char)((colNum & 0x00FF0000) >> 16);
   pData[i++] = (char)((colNum & 0x0000FF00) >> 8);
   pData[i++] = (char)((colNum & 0x000000FF));

   pData[i++] = (char)((ecc & 0xFF000000) >> 24);
   pData[i++] = (char)((ecc & 0x00FF0000) >> 16);
   pData[i++] = (char)((ecc & 0x0000FF00) >> 8);
   pData[i++] = (char)((ecc & 0x000000FF));

   pData[i++] = (char)((xScale & 0xFF000000) >> 24);
   pData[i++] = (char)((xScale & 0x00FF0000) >> 16);
   pData[i++] = (char)((xScale & 0x0000FF00) >> 8);
   pData[i++] = (char)((xScale & 0x000000FF));

   pData[i++] = (char)((yScale & 0xFF000000) >> 24);
   pData[i++] = (char)((yScale & 0x00FF0000) >> 16);
   pData[i++] = (char)((yScale & 0x0000FF00) >> 8);
   pData[i++] = (char)((yScale & 0x000000FF));

   return pData;
}

void OZPdf417Comp::readMe(const char* pData, const int length){
// serialize functin
// pData is result of writeMe() function call
// length is result of writeMe() function call
// length is binary length
  if(length != 20){
    throw new CException(0);
  }
  int i = 0;
  rowNum = ((pData[i++] << 24) + (pData[i++] << 16) + (pData[i++] << 8) +
            (pData[i++] << 0));
```

```
  colNum = ((pData[i++] << 24) + (pData[i++] << 16) + (pData[i++] << 8) +
            (pData[i++] << 0));
  ecc = ((pData[i++] << 24) + (pData[i++] << 16) + (pData[i++] << 8) +
            (pData[i++] << 0));
  xScale = ((pData[i++] << 24) + (pData[i++] << 16) + (pData[i++] << 8) +
            (pData[i++] << 0));
  yScale = ((pData[i++] << 24) + (pData[i++] << 16) + (pData[i++] << 8) +
            (pData[i++] << 0));
}

void OZPdf417Comp::paint(HDC hDC, CString data, const float x, const float y,
const float w, const float h, const float scale){
// hDC is device context handle
// data is component caption
// x and y are component pixel positon
// w and h are component pixel size
// scale is extension ratio. (1.0f 1.5f 2.0f etc...)
//(int)(x*scale) and (int)(y*scale) are real position in HDC

  LPPICTURE gpPicture =
                  EncodePrint(data, rowNum,colNum,ecc,xScale,yScale);//함수호출
  if(gpPicture == NULL) return;
  long hmWidth;
  long hmHeight;
  gpPicture->get_Width(&hmWidth);
  gpPicture->get_Height(&hmHeight);
  // convert himetric to pixelsA
  int imageWidth;
  int imageHeight;
  imageWidth= MulDiv(hmWidth*72, GetDeviceCaps(hDC, LOGPIXELSX), 2540*96);
  imageHeight= MulDiv(hmHeight*72, GetDeviceCaps(hDC, LOGPIXELSY), 2540*96);
  imageWidth = (int)(imageWidth*scale);
  imageHeight = (int)(imageHeight*scale);
  RECT rc = { 0, 0, imageWidth, imageHeight };
  HRESULT hesult = gpPicture->Render(hDC, (int)(x*scale), (int)(y*scale),
                imageWidth, imageHeight, 0, hmHeight, hmWidth, -hmHeight, &rc);

  gpPicture->Release();
}

void OZPdf417Comp::print(HDC hDC, CString data, const float x, const float y,
const float w, const float h, const float scale, const float x_offset,
const float y_offset){
// hDC is device context handle
// data is component caption
// x and y are component pixel positon
// w and h are component pixel size
// scale is extension ratio.
```

```
// x_offset and y_offset are print offset
//(int)(x*scale+x_offset) and (int)(y*scale+y_offset) are real position in HDC

  LPPICTURE gpPicture =
                  EncodePrint(data, rowNum,colNum,ecc,xScale,yScale);//함수호출
  if(gpPicture == NULL) return;
  long hmWidth;
  long hmHeight;
  gpPicture->get_Width(&hmWidth);
  gpPicture->get_Height(&hmHeight);
  // convert himetric to pixelsA
  int imageWidth;
  int imageHeight;
  imageWidth= MulDiv(hmWidth, GetDeviceCaps(hDC, LOGPIXELSX), 2540);
  imageHeight= MulDiv(hmHeight, GetDeviceCaps(hDC, LOGPIXELSY), 2540);
  RECT rc = { 0, 0, imageWidth, imageHeight };
  HRESULT hesult = gpPicture->Render(hDC, (int)(x*scale+x_offset),
                  (int)(y*scale+y_offset),
                  imageWidth, imageHeight, 0, hmHeight, hmWidth, -hmHeight,
                  &rc);
  gpPicture->Release();
}

void OZPdf417Comp::getAutosize(HDC hDC, CString data, float * w, float * h){
// hDC is device context handle
// data is component caption
// *w and *h are component pixel size
// prefered size may set to w and h  (*w = 100; *h = 150;)

  LPPICTURE gpPicture =
                  EncodePrint(data, rowNum,colNum,ecc,xScale,yScale);//함수호출
  if(gpPicture == NULL) return;
  long hmWidth;
  long hmHeight;
  gpPicture->get_Width(&hmWidth);
  gpPicture->get_Height(&hmHeight);
  // convert himetric to pixelsA
  *w = (float)MulDiv(hmWidth*72, GetDeviceCaps(hDC, LOGPIXELSX), 2540*96);
  *h = (float)MulDiv(hmHeight*72, GetDeviceCaps(hDC, LOGPIXELSY), 2540*96);
  gpPicture->Release();
}
```

Appendix 1. SchedulerCom 활용하기

본 장에서는 Scheduler API 중 makePDF 함수와 export 함수를 ASP에서 사용할 수 있도록 하는 SchedulerCom의 함수와 makePDF와 export 함수를 사용하는 ASP 샘플에 대해 설명합니다.

SchedulerCom 함수

■ **Com 초기화 및 해제 함수**

- .Init

Prototype	`.Init()`
Definition	COM을 초기화합니다.
Argument	없음

- .Clean

Prototype	`.Clean()`
Definition	COM의 설정 정보를 해제시킵니다.
Argument	없음

■ **오즈 서버 설정 함수**

- .SetServerType

Prototype	`.SetServerType String ServerType`		
Definition	오즈 서버의 타입을 설정합니다.		
Argument	*ServerType*	*"TCP"*	오즈 서버가 데몬 타입일 경우 "TCP"로 설정
		"Servlet"	오즈 서버가 서블릿 타입일 경우 "Servlet"으로 설정 (기본값)

- .SetServerIP

Prototype	`.SetServerIP String IP`	
Definition	오즈 서버가 데몬 타입일 경우 오즈 서버의 IP를 설정합니다.	
Argument	*IP*	오즈 서버의 IP ex) "127.0.0.1"

- .SetServerPort

Prototype	`.SetServerPort String Port`	
Definition	오즈 서버가 데몬 타입일 경우 오즈 서버의 Port 번호를 설정합니다.	
Argument	*Port*	오즈 서버의 Port 번호 ex) "8003"

- .SetServerURL

Prototype	`.SetServerURL String URL`	
Definition	오즈 서버가 서블릿 타입일 경우 오즈 서버의 URL을 설정합니다.	
Argument	*URL*	오즈 서버 URL ex) "http://localhost:7001/oz/server"

■ **오즈 스케줄러 서버 설정 함수**

- .SetSchedulerIP

Prototype	`.SetSchedulerIP String IP`	
Definition	오즈 스케줄러 서버의 IP를 설정합니다.	
Argument	*IP*	오즈 스케줄러 서버의 IP ex) "127.0.0.1"

- .SetSchedulerPort

Prototype	`.SetSchedulerPort String Port`	
Definition	오즈 스케줄러 서버의 Port 번호를 설정합니다.	
Argument	*Port*	오즈 스케줄러 서버의 Port 번호 ex) "9521"

■ **사용자 정보 설정 함수**

- .SetUser

Prototype	`.SetUser String UserID`	
Definition	접속할 사용자의 아이디를 설정합니다.	
Argument	*UserID*	사용자 아이디 ex) "admin"

- .SetPassword

Prototype	`.SetPassword String Password`	
Definition	접속할 사용자의 패스워드를 설정합니다.	
Argument	*Password*	사용자 패스워드 ex) "admin"

■ **Key 설정 함수**

- .SetProperty

Prototype	`.SetProperty String Key, String Value`	
Definition	보고서 파일명, 패러미터 등을 설정합니다.	
Argument	*Key*	설정할 Key
	Value	설정할 값

※ 참고사항 : ".SetProperty"에서 사용할 수 있는 Key와 Value

Key	Value	설명
"launch_type"	"Immediately"	Scheduler에서 Task로 인식하기 위해 필요하며, 값은 항상 "Immediately"로 설정하여야 합니다.
"report_name"	"파일명"	익스포트할 보고서(*.ozr) 파일명을 설정합니다.
"category_name"	"/<Category>"	익스포트할 보고서의 파일 카테고리를 설정합니다.
"export.confirmsave"	"false"	항상 "false"로 설정합니다.
"parameter_count"	"패러미터 개수"	폼 패러미터 개수와 ODI 패러미터 개수를 더한 전체 패러미터 개수를 설정합니다. ※ 주의사항 : 패러미터가 없을 경우 "0"으로 설정합니다.
"parameter_name_<index>"	"패러미터명"	패러미터명을 설정합니다. 폼 패러미터의 경우에는 "[FORM].패러미터명"으로 설정하고 ODI 패러미터의 경우에는 "ODI명.패러미터명"으로 설정합니다.
"parameter_value_<index>"	"패러미터 값"	패러미터 값을 설정합니다.

※ 주의사항 : ".SetProperty" 함수를 이용하여 설정되는 Key 중 "launch_type", "report_name", "category_name", "export.confirmsave", "parameter_count"은 반드시 설정하여야 제대로 동작됩니다.

- .SetExportProperty

Prototype	.SetExportProperty String Key, String Value	
Definition	익스포트할 보고서 정보 또는 익스포트 파일 등 에 관한 정보를 설정합니다.	
Argument	*Key*	설정할 Key
	Value	설정할 값

※ 참고사항 : ".SetExportProperty"에서 사용할 수 있는 Key와 Value

Key	Value	설명
"connection.server"	"서버 IP"	오즈 서버가 데몬 타입일 경우 오즈 서버의 IP를 설정합니다.
"connection.port"	"서버 Port"	오즈 서버가 데몬 타입일 경우 오즈 서버의 Port 번호를 설정합니다.
"connection.servlet"	"서버 URL"	오즈 서버가 서블릿 타입일 경우 오즈 서버의 URL을 설정합니다.
"connection.reportName"	"보고서 파일명"	익스포트할 보고서의 파일명을 설정합니다.
"connection.fetchtype"	"BATCH"	패치 타입을 설정합니다.
"connection.pcount"	"패러미터 개수"	폼 패러미터 개수를 설정합니다.
"connection.args<index>=패러미터명"	"패러미터 값"	폼 패러미터 값을 설정합니다.
"odi.odinames"	"ODI명,ODI명,.."	사용되는 ODI명을 설정합니다. 여러 개의 ODI인 경우 각각의 ODI명을 콤마(",")로 구별하여 나열합니다.
"odi.ODI명.pcount",	"패러미터 개수"	ODI 패러미터 개수를 설정합니다.
"odi.ODI명.args<index>"	"패러미터명=패러미터값"	패러미터명과 패러미터 값을 설정합니다.

"export.format"	"익스포트할 파일 확장자"	익스포트할 파일의 확장자를 "/"로 구분하여 나열합니다. "ozd/html/jpg/xls/doc/svg/txt/ppt/tif/csv"
"<ozd/html/jpg/xls/doc/svg/txt/ppt/tif/csv>.filename"	"파일명"	익스포트할 파일명을 설정합니다. ex).setExportProperty "ozd.filename", "test.ozd"
"viewer.childcount"	"자식 개수"	멀티 보고서의 경우 자식 보고서의 개수를 설정합니다.

※ 참고사항 : 자식 보고서의 경우 부모 보고서에서 설정할 수 있는 Key에 "child<index>."을 조합하여 사용할 수 있습니다.

ex) "child1.connection.server"

ex) "child1.connection.port"

■ **익스포트 함수**

- .MakePDF

Prototype	`.MakePDF(String ExportType)`		
Definition	pdf 파일로 익스포트합니다. ※ 주의사항 : 서버에 설정된 스케줄러의 환경 설정 중 "ViewType"을 "None"으로 설정한 경우에는 사용할 수 없습니다.		
Argument	*ExportType*	*"SHOW"*	사용자 PC에 설치되어 있는 PDF Reader에 익스포트한 pdf 파일을 보여줍니다.
		"ATTACH"	사용자가 pdf 파일을 다운로드 받을 수 있는 창이 뜹니다.
		"NONE"	pdf 파일을 익스포트만 하고, 사용자에게는 아무런 액션을 보내지 않습니다.

- .Export

Prototype	`.Export()`
Definition	pdf, ozd, html, jpg, xls, doc, svg, txt, ppt, tif, csv 파일로 익스포트합니다. ※ 주의사항 : 서버에 설정된 스케줄러의 환경 설정 중 "ViewType"을 "None"으로 설정한 경우에는 사용할 수 없습니다.
Argument	없음

- .IsExportSuccessed

Prototype	`.IsExportSuccessed()`	
Definition	익스포트 결과를 리턴합니다.	
Return	*true*	익스포트 성공
	false	익스포트 실패

■ **기타 함수**

- .ShowMessage

Prototype	`.ShowMessage(String Msg)`	
Definition	설정한 메시지를 표시합니다.	
Argument	*Msg*	표시할 메시지

SchedulerCom 주의사항

■ **사용 가능 서버**

"SchedulerCom"은 MS 계열인 서버에서만 사용할 수 있습니다.

■ **SchedulerCOM.dll 파일 레지스트리에 등록**

"SchedulerCom"을 사용하기 위해서는 "SchedulerCOM.dll" 파일을 서버의 레지스트리에 등록하여야 합니다.

만일 "SchedulerCOM.dll" 파일이 "C:\OZServer\Scheduler\"에 있을 경우 실행창에

```
regsvr32  " C:\OZServer\Scheduler\SchedulerCOM.dll"
```

로 입력하여 등록합니다.

■ **익스포트시 파일 경로**

MakePDF 또는 Export 함수를 이용하여 보고서 파일을 익스포트하면 익스포트된 파일의 이름은 설정할 수 있으나 익스포트 경로는 패러미터로 설정할 수 없고, 스케줄러에 설정되어 있는 리파지토리 경로로 자동 지정됩니다. 스케줄러에 설정되어 있는 리파지토리 경로는 "scheduler_server.properties"의 "RepositoryFileRootPath"에 설정된 값을 참조하시기 바랍니다.

MakePDF 함수 사용 ASP 샘플

```
<%
Dim Com
Set Com = Server.CreateObject("SchedulerCOM.CSchedulerCall.1")

If Not IsObject (Com) Then

   Response.Write("PDF File이 정상적으로 생성되지 않았습니다.")
   Response.End

Else

   With Com

      'Init. COM
         .Init()
      ' TCP-Daemon Type Server
         .SetServerType "TCP"
         .SetServerIP "127.0.0.1"
         .SetServerPort "8003"

      ' Servlet Type Server
      '   .SetServerType "Servlet"
      '   .SetServerURL "http://localhost:7001/oz/server"

      ' Set Scheduler Info.
         .SetSchedulerIP "127.0.0.1"
         .SetSchedulerPort "9521"

      ' set User Info.
         .SetUser "admin"
         .SetPassword "admin"

      ' set Launch Type
         .SetProperty "launch_type", "Immediately"

      ' set Report Info.
         .SetProperty "report_name", "parameter_test.ozr"
         .SetProperty "category_name", "/"
         .SetProperty "export.confirmsave", "false"

      ' with NO parameter.
         .setProperty "parameter_count", "0"
```

```
      ' with parameters.
      ' .setProperty "parameter_count", "4"
      ' .setProperty "parameter_name_1", "[FORM].formparam1"
      ' .setProperty "parameter_value_1", "PARAM 1"
      ' .setProperty "parameter_name_2", "[FORM].formparam2"
      ' .setProperty "parameter_value_2", "PARAM 2"
      ' .setProperty "parameter_name_3", "parameter_test.odiparam1"
      ' .setProperty "parameter_value_3", "PARAM 3"
      ' .setProperty "parameter_name_4", "parameter_test.odiparam2"
      ' .setProperty "parameter_value_4", "PARAM 4"

       ' set Export Info.
          .setExportProperty "pdf.filename", "PDF_TEST.pdf"

      .MakePDF("NONE")

      res = .IsExportSuccessed()
      if res = "true" then
          .ShowMessage ("successed...")
      else
          .ShowMessage ("failed...")
      end if

      .Clean()

   End With

   Set Com = Nothing

End If
%>
```

Export 함수 사용 ASP 샘플

```
<%
Dim Com
Set Com = Server.CreateObject("SchedulerCOM.CSchedulerCall.1")

If Not IsObject (Com) Then
   Response.Write("File이 정상적으로 생성되지 않았습니다.")
```

```
    Response.End

Else
    With Com
        .Init()
        .SetServerType "TCP"
        .SetServerIP "127.0.0.1"
        .SetServerPort "8003"
        .SetSchedulerIP "127.0.0.1"
        .SetSchedulerPort "9521"
        .SetUser "admin"
        .SetPassword "admin"
        .SetProperty "launch_type", "Immediately"
        .setProperty "cfg.type","new"

        .setExportProperty "connection.server", "127.0.0.1"
        .setExportProperty "connection.port", "8003"
        .setExportProperty "connection.reportName", "/parameter_test.ozr"
        .setExportProperty "connection.fetchtype", "BATCH"
        .setExportProperty "connection.pcount", "2"
        .setExportProperty "connection.args1=formparam1", "form1"
        .setExportProperty "connection.args2=formparam2", "form2"

        .setExportProperty "odi.parameter_test.args1", "odiparam1=odi1"
        .setExportProperty "odi.parameter_test.args2", "odiparam2=odi2"
        .setExportProperty "odi.parameter_test.pcount", "2"
        .setExportProperty "odi.odinames", "parameter_test"

        .setExportProperty "export.format",
                           "ozd/html/jpg/xls/doc/svg/txt/ppt/tif/csv"
        .setExportProperty "ozd.filename", "test.ozd"
        .setExportProperty "html.filename", "test.html"
        .setExportProperty "jpg.filename", "test.jpg"
        .setExportProperty "excel.filename", "test.xls"
        .setExportProperty "word.filename", "test.doc"
        .setExportProperty "svg.filename", "test.svg"
        .setExportProperty "text.filename", "test.txt"
        .setExportProperty "ppt.filename", "test.ppt"
        .setExportProperty "tiff.filename", "test.tif"
        .setExportProperty "csv.filename", "test.csv"

        .setExportProperty "viewer.childcount", "1"
        .setExportProperty "child1.connection.server", "127.0.0.1"
        .setExportProperty "child1.connection.port", "8003"
```

```
      .setExportProperty "child1.connection.reportName",
                         "/parameter_test.ozr"
      .setExportProperty "child1.connection.fetchtype", "BATCH"
      .setExportProperty "child1.connection.pcount", "2"
      .setExportProperty "child1.connection.args1=formparam1", "form1"
      .setExportProperty "child1.connection.args2=formparam2", "form2"

      .setExportProperty "child1.odi.parameter_test.args1", "odiparam1=odi1"
      .setExportProperty "child1.odi.parameter_test.args2", "odiparam2=odi2"
      .setExportProperty "child1.odi.parameter_test.pcount", "2"
      .setExportProperty "child1.odi.odinames", "parameter_test"

      .setExportProperty "child1.export.format",
                          "ozd/html/jpg/xls/doc/svg/txt/ppt/tif/csv"
      .setExportProperty "child1.ozd.filename", "child_test.ozd"
      .setExportProperty "child1.html.filename", "child_test.html"
      .setExportProperty "child1.jpg.filename", "child_test.jpg"
      .setExportProperty "child1.excel.filename", "child_test.xls"
      .setExportProperty "child1.word.filename", "child_test.doc"
      .setExportProperty "child1.svg.filename", "child_test.svg"
      .setExportProperty "child1.text.filename", "child_test.txt"
      .setExportProperty "child1.ppt.filename", "child_test.ppt"
      .setExportProperty "child1.tiff.filename", "child_test.tif"
      .setExportProperty "child1.csv.filename", "child_test.csv"

      .Export()

      Response.Write(.IsExportSuccessed())

      .Clean()

   End With

   Set Com = Nothing

End If
%>
```

Appendix 2. Servlet API를 이용하여 다른 서버와 데이터 통신하기

본 장에서는 Servlet API를 이용하여 데이터 관련 통신을 오즈 서버와 하지 않고, 다른 서버와 연동하여 통신하는 방식에 대해 설명합니다.

Servlet API(ozsdmapi.jar) 연동 개요

Servlet API를 연동하는 방법은 기존 방식과 다르게 데이터 관련 통신을 오즈 서버와 하지 않고, 사용자가 지정한 특정 서버와 연동하여 생성된 데이터를 가져옵니다. 단, OZR, OZA 등의 폼 파일은 오즈 서버에서 가져옵니다.

특정 URL로부터 데이터를 오즈 뷰어로 받아오는 경우 POST 패러미터로 ODI 패러미터 등의 정보를 주어 오즈 데이터 포맷으로 데이터를 받습니다.

오즈 애플리케이션에서 행 삽입, 삭제, 변경 쿼리 등으로 DataAction을 하는 경우 해당 DataAction 내용이 POST 패러미터로 전달하여 URL을 호출하고, DataAction이 성공한 경우에는 "ok" 또는 "success"로 결과를 리턴받을 수 있습니다.

Servlet API를 이용할 경우에는 ODI의 DB 관련 정보는 사용되지 않기 때문에 사용자 데이터(UDS) 스토어를 이용하고, 데이터셋에 사용자 정의 필드를 추가하여 수동 정의하여야 합니다.

POST 패러미터

클라이언트로 데이터를 전송하거나 클라이언트로부터 행 삽입, 삭제, 변경 정보를 받아 처리할 경우 POST 패러미터를 이용하거나 POST 패러미터에서 정보를 추출하여 처리합니다.

■ **서버에 데이터를 조회할 때 서버로 보내지는 POST 패러미터**

패러미터 이름	설명
_OZ_ODIFetchType_	Fetch 타입 DM_BATCH_FETCH 또는 DM_CONCURRENT_FETCH FetchUnit이 DM_PER_DATASET인 경우에 Fetch 타입은 항상 DM_CONCURRENT_FETCH임.
_OZ_ODIITEM_	ODI 파일명
_OZ_ODICATEGORY_	ODI 카테고리명

_OZ_DATASET_	데이터셋명 FetchUnit이 DM_PER_DATASET인 경우에만 전송됨.
MyParam	ODI 패러미터들이 같은 이름으로 전송됨.

■ **서버에 DataAction 내용을 보낼 때 서버로 보내지는 POST 패러미터**

패러미터 이름	설명
MyParam	odi 패러미터들이 같은 이름으로 전송됨.
_OZ_DAC_CNT	DataAction의 개수로 <INDEX>와 연관됨.
<INDEX>.DATASET	DataAction의 데이터셋명
<INDEX>.TYPE	DataAction의 CUD 타입 Insert, Delete, RowUpdate 중 하나
<INDEX>.EXT	DataAction의 Extra 값
<INDEX>.SRC_CNT	DataAction의 Source 항목 개수로 <INDEX2>와 연관됨.
<INDEX>.SF_<INDEX2>	DataAction의 <INDEX2>번째 SourceName
<INDEX>.SV_<INDEX2>	DataAction의 <INDEX2>번째 SourceValue
<INDEX>.TRG_CNT	DataAction의 Target 항목 개수로 <INDEX2>와 연관됨.
<INDEX>.DF_<INDEX2>	DataAction의 <INDEX2>번째 TargetName
<INDEX>.DV_<INDEX2>	DataAction의 <INDEX2>번째 TargetValue

※ 참고사항 : 3개의 ODI 패러미터가 정의되어 있고, DataAction 3개를 Commit한 경우 전송되는 POST 패러미터의 예

```
paramA=somevalue
paramB=somevalue
paramC=somevalue
_OZ_DAC_CNT=3
0.DATASET=dataset1
0.TYPE=Insert
0.EXT=
0.SRC_CNT=2
0.SF_0=FieldName1
0.SV_0=a
0.SF_1=FieldName2
0.SV_1=b
1.DATASET=dataset1
1.TYPE=Delete
1.EXT=
1.TRG_CNT=1
1.DF_0=FieldName1
```

```
1.DV_0=a
2.DATASET=dataset1
2.TYPE=RowUpdate
2.EXT=
2.SRC_CNT=2
2.SF_0=FieldName1
2.SV_0=newvalue
2.SF_1=FieldName2
2.SV_1=oldvalue
2.TRG_CNT=2
2.DF_0=FieldName1
2.DV_0=newvalue
2.DF_1=FieldName2
2.DV_1=oldvalue
```

Servlet API 함수

■ **DataModuleFactory 클래스 함수**

- getDataModule

Prototype	`public static DataModule getDataModule(String FetchType) throws OZSDMException`	
Definition	해당 Fetch 타입의 데이터 모듈을 가져옵니다.	
Argument	*FetchType*	Fetch 타입 DM_CONCURRENT_FETCH 또는 DM_BATCH_FETCH

■ **DataModule 클래스 함수**

- init

Prototype	`public void init(OutputStream out) throws IOException`	
Definition	데이터 모듈 바인딩을 설정한 Stream으로 초기화합니다.	
Argument	*out*	Response의 OutputStream

- startBindding

Prototype	`public void startBinding() throws IOException, SQLException`

Definition	데이터 모듈의 바인딩을 시작합니다. 바인딩 시작 전에 반드시 호출해야 합니다.

- endBindding

Prototype	`public void endBinding() throws IOException, SQLException`
Definition	데이터 모듈의 바인딩을 종료합니다. 바인딩이 종료된 후 반드시 호출해야 합니다.

- startSet

Prototype	`public void startSet(String DatasetName) throws IOException`
Definition	해당 데이터셋의 바인딩을 시작합니다. 데이터셋 시작 전에 반드시 호출해야 합니다.
Argument	*DatasetName* 데이터셋 이름

- endSet

Prototype	`public void endSet(String DatasetName) throws IOException`
Definition	해당 데이터셋의 바인딩을 종료합니다.
Argument	*DatasetName* 데이터셋 이름

- makeSDM_SET

Prototype	`public void makeSDM_SET(String DatasetName, ResultSet rs, OutputStream out) throws OZSDMException, IOException`	
Definition	DM_PER_DATASET 단위로 바인딩하여 SDM을 만듭니다.	
Argument	*DatasetName*	데이터셋 이름
	rs	ResultSet
	out	Response의 OutputStream

- addParameter

Prototype	`public void addParameter(String ParamName, int FieldType, Object Value)`	
Definition	패러미터 값을 설정합니다.	
Argument	*ParamName*	패러미터 이름
	FieldType	패러미터 필드 타입

	Value	설정할 패러미터 값

- addSetInfo

Prototype	public void addSetInfo(String DatasetName, String MasterDatasetName, String[] FieldNames, int[] FieldTypes) throws IllegalArgumentException	
Definition	데이터 모듈에 해당되는 데이터셋을 설정합니다.	
Argument	*DatasetName*	데이터셋 이름
	MasterDatasetName	데이터셋의 마스터 데이터셋 이름 단, 마스터 데이터셋이 없는 경우 ""로 설정
	FieldNames	데이터셋의 필드 이름
	FieldTypes	데이터셋의 필드의 타입

- addrow

Prototype	public void addRow(String DatasetName, HashMap hMap) throws IOException, SQLException	
Definition	ResultSet의 Row을 설정한 데이터셋에 추가합니다.	
Argument	*DatasetName*	데이터셋 이름
	hMap	HashMap 형태로 지정된 추가할 Row 내용 예) ... rs = stmt.executeQuery(query); while(rs.next()) { HashMap hMap = new HashMap(); hMap.put(_FN[0], rs.getString(_FN[0])); hMap.put(_FN[1], rs.getString(_FN[1])); hMap.put(_FN[2], rs.getString(_FN[2])); hMap.put(_FN[3], rs.getString(_FN[3])); hMap.put(_FN[4], rs.getString(_FN[4])); hMap.put(_FN[5], rs.getString(_FN[5])); module.addRow("SET_1", hMap); } ...

- addrow

Prototype	`public void addRow(String DatasetName, List list) throws IOException, SQLException`	
Definition	ResultSet의 Row을 설정한 데이터셋에 추가합니다.	
Argument	*DatasetName*	데이터셋 이름
	list	ArrayList 형태로 지정된 추가할 Row 내용

- addrow

Prototype	`public void addRow(String DatasetName, String[] arr) throws IOException, SQLException`	
Definition	ResultSet의 Row을 설정한 데이터셋에 추가합니다.	
Argument	*DatasetName*	데이터셋 이름
	arr	String[] 형태로 지정된 추가할 Row 내용

- sendBindErrorMessage

Prototype	`public void sendBindErrorMessage(String msg) throws IOException`	
Definition	바인딩시에 에러 발생시 에러를 처리합니다. 바인딩시 에러가 발생하면 반드시 메세지를 처리해줘야 하는 메소드로, 클라이언트로 에러 메시지를 전송합니다.	
Argument	*msg*	에러 메시지

- sendErrorMessage

Prototype	`public void sendErrorMessage(String msg, OutputStream out) throws IOException`	
Definition	바인딩시 외의 에러 발생시 에러를 처리합니다. 바인딩시 외의 에러가 발생하면 반드시 메세지를 처리해줘야 하는 메소드로, 클라이언트로 에러 메시지를 전송합니다.	
Argument	*msg*	에러 메시지
	out	Response의 OutputStream

Servlet API 예 1 : Application - 데이터 검색, DataAction

Servlet API를 이용하여 오즈 애플리케이션의 Table을 이용하여 데이터를 표시하고, 데이터를 삽입, 삭제, 변경하는 예제를 살펴보겠습니다

Servlet API를 이용하기 위해서는 먼저 ozsdmapi.jar 파일을 웹 서버에 등록하고, 데이터를 검색하는 클래스 파일과 DataAction을 실행하는 클래스 파일을 만들어 웹 서버에 등록한 후 쿼리 디자이너를 이용하여 UDS 스토어의 데이터셋에 행 삽입, 삭제, 변경 쿼리를 입력한 ODI 파일을 만들어 오즈 애플리케이션의 프로젝트에 추가하여야 합니다.

Step 1 ozsdmapi.jar 파일을 웹 서버에 등록하기

Servlet API 연동 모듈인 ozsdmapi.jar 파일을 웹 서버의 라이브러리 폴더에 복사하여 등록하였습니다. 본 예제에서는 Tomcat 5.0의 webapps\ROOT\WEB-INF\lib 폴더에 ozsdmapi.jar 파일을 복사하였습니다.

Step 2 데이터 검색용 클래스 파일 만들기

데이터를 검색하는 클래스 파일을 만듭니다. 본 예제에서는 아래와 같은 java 소스 파일을 이용하여 DataModuleSampleServlet.class 파일로 만들었습니다.

```
package sample;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.*;

import java.util.*;

import oz.framework.api.DataModule;
import oz.uds.rs.ListMapResultSet;
import oz.sdm.DataModuleFactory;
import oz.uds.rs.ListMapResultSet;

/**
 * <p>Title: OZ SDM API</p>
 * <p>Description: </p>
 * <p>Copyright: Copyright (c) 2005</p>
 * <p>Company: </p>
 * @author Forcs
 * @version 1.0
 */

public class DataModuleSampleServlet extends HttpServlet
{
    private static final String _KEY_ODI_FETCH_TYPE = "_OZ_ODIFetchType_";
    private static final String _KEY_ODIITEM       = "_OZ_ODIITEM_";
    private static final String _KEY_ODICATEGORY    = "_OZ_ODICATEGORY_";
    private static final String _KEY_OZ_DATASET_    = "_OZ_DATASET_";

    private Connection m_conn = null;

    public void init(ServletConfig config)
        throws ServletException
    {
```

```
        super.init(config);


    }

    public  void  doPost(HttpServletRequest  request,  HttpServletResponse
response)
        throws ServletException, IOException
    {
        process(request, response);
    }

    public  void  doGet(HttpServletRequest  request,  HttpServletResponse
response)
        throws ServletException, IOException
    {
        process(request, response);
    }

    private  void  process(HttpServletRequest  request,  HttpServletResponse
response)
        throws ServletException, IOException
    {

        // init할때 커낵션을 정보를 설정하여 connection을 가져온다.
        try {
            String _URL = "jdbc:odbc:ozdemokr30";
            Properties prop = new Properties();
            prop.put("user", "");
            prop.put("password", "");

            Driver              driver              =              (Driver)
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver").newInstance();
            m_conn = driver.connect(_URL, prop);
            String aaa = null;
        }
        catch(Exception e) {
            DataModule.sendErrorMessage("connection
error...",response.getOutputStream());
            throw new ServletException(e.getMessage());
        }

        try {
            System.out.println("--------------------");
            Enumeration enum = request.getParameterNames();
            while (enum.hasMoreElements()) {
                String temp = (String) enum.nextElement();
                System.out.println("name=" + temp + "   value=" +
                                getEncode(request.getParameter(temp)));
            }

            //  ------------------------------------------------------------
---------
            // fetch parameters
            // "DM_BATCH_FETCH|DM_CONCURRENT_FETCH"
            String odiFetchType = getEncode(request.getParameter(
                _KEY_ODI_FETCH_TYPE));
            String item = getEncode(request.getParameter(_KEY_ODIITEM));
            String                          category                        =
getEncode(request.getParameter(_KEY_ODICATEGORY));
            String                           dataset                        =
getEncode(request.getParameter(_KEY_OZ_DATASET_));
            //  ------------------------------------------------------------
---------

            if (odiFetchType == null) odiFetchType = "DM_CONCURRENT_FETCH";

            if ( (dataset == null) || (dataset.length() == 0)) {
                // ODI FetchUnit이 DM_PER_DATAMODULE 방식이다.
                if (item.equals("JASMIN_SAMPLE3.odi")) {
                    // Master-Detail간 Sample이다.
                    Master_Detail_Style(request, response, odiFetchType);
                }
                else if (item.equals("JASMIN_SAMPLE2.odi") ||
                        (item.equals("JASMIN_SAMPLE4.odi"))) {
                    // 일반적인 SET하나인 Sample이다.
                    System.out.println("default...");
```

```
                default_Style(request, response, odiFetchType);
            }
            else {
                DataModule.sendErrorMessage("unsupported odi error...",
                                            response.getOutputStream());
                throw new ServletException("Unknown Item :" + item);
            }

         }
         else {
            // ODI FetchUnit이 DM_PER_DATASET 방식이다.
            // 이 방식일 경우에는 odiFetchType은 무조건 DM_CONCURRENT_FETCH
이어야 한다.
            // 아닐 경우 지원하지 않는다.
            if (odiFetchType.equalsIgnoreCase("DM_CONCURRENT_FETCH")) {
                this.SET_Style(request, response, "DM_CONCURRENT_FETCH");
            }
            else {
                DataModule.sendErrorMessage("unsupported error...",
                                            response.getOutputStream());
                throw new ServletException("unsupported");
            }
         }

         if (m_conn != null) {
            try {
                m_conn.close();
            }
            catch (Exception e) {
            }
         }
      }
      catch(Exception ex){
         // 바인딩 외의 에러는 모두 sendErrorMessage메소드로 통해서 전송한후
         // throw를 해줘야 한다. 그렇지 않으면 client가 다운될 수 있음
         DataModule.sendErrorMessage(ex.getMessage(),
                                     response.getOutputStream());
         throw new ServletException(ex.getMessage());
      }

   }

   private    void    Master_Detail_Style(HttpServletRequest    request,
HttpServletResponse response, String fetchType)
      throws ServletException, IOException
   {
      Statement stmt1 = null;
      Statement stmt4 = null;
      ResultSet rs1 = null;
      ResultSet rs4 = null;

      DataModule module = null;

         // jasmin/JASMIN_SAMPLE1.odi
         // SET_1
         String[] _FN1 = {"FirstName" };
         int[]   _FT1 = {java.sql.Types.VARCHAR };
         // SET_4
         String[] _FN4 = {"ContactId", "FirstName", "LastName", "Phone",
"Fax", "Email", "OrgUnitId", "UserName" };
         int[]   _FT4 = {4, 12, 12, 12, 12, 12, 5, 12 };
      try {
         module = DataModuleFactory.getDataModule(fetchType);

         module.init(response.getOutputStream());

         // set SET Info.
         module.addSetInfo("SET_1", "", _FN1, _FT1);
         module.addSetInfo("SET_4", "SET_1", _FN4, _FT4);
      }
      catch(Exception ex) {
         throw new ServletException(ex.getMessage());
      }
      try {
         module.startBinding();
```

```
            String query1 = "select distinct FirstName from contact";

            module.startSet("SET_1");
            stmt1 = m_conn.createStatement();
            rs1 = stmt1.executeQuery(query1);
            while(rs1.next()) {
               String f1 = rs1.getString(_FN1[0]);
               HashMap map1 = new HashMap();
               map1.put(_FN1[0], f1);

               module.addRow("SET_1", map1);

               String query4 = "select * from contact where FirstName='" +
f1 + "' order by contactid";

               module.startSet("SET_4");
               stmt4 = m_conn.createStatement();
               rs4 = stmt4.executeQuery(query4);
               while(rs4.next()) {
                  HashMap map4 = new HashMap();
                  map4.put(_FN4[0], new Integer(rs4.getInt(_FN4[0])));
                  map4.put(_FN4[1], rs4.getString(_FN4[1]));
                  map4.put(_FN4[2], rs4.getString(_FN4[2]));
                  map4.put(_FN4[3], rs4.getString(_FN4[3]));
                  map4.put(_FN4[4], rs4.getString(_FN4[4]));
                  map4.put(_FN4[5], rs4.getString(_FN4[5]));
                  map4.put(_FN4[6], new Integer(rs4.getInt(_FN4[6])));
                  map4.put(_FN4[7], rs4.getString(_FN4[7]));

                  module.addRow("SET_4", map4);
               }
               rs4.close(); rs4 = null;
               stmt4.close(); stmt4 = null;

               module.endSet("SET_4");
            }

            rs1.close(); rs1 = null;
            stmt1.close(); stmt1 = null;

            module.endSet("SET_1");

            module.endBinding();
         }
         catch(Exception e) {
            e.printStackTrace();
            // 바인딩시 에러 전송은 sendBindErrorMessage메소드로 꼭 해야 한다.
            if(module != null) {
               module.sendBindErrorMessage(e.toString());
            }
         }
         finally {

            if(rs1 != null) {
               try { rs1.close(); } catch(Exception e) {}
            }
            if(stmt1 != null) {
               try { stmt1.close(); } catch(Exception e) {}
            }
            if(rs4 != null) {
               try { rs4.close(); } catch(Exception e) {}
            }
            if(stmt4 != null) {
               try { stmt4.close(); } catch(Exception e) {}
            }
         }
      }

      private      void      default_Style(HttpServletRequest      request,
HttpServletResponse response, String fetchType)
         throws ServletException, IOException
      {
         System.out.println("select time=="+System.currentTimeMillis());
         Statement stmt = null;
         ResultSet rs = null;

         DataModule module = null;
```

```
        // jasmin/JASMIN_SAMPLE2.odi
        // SET_1
        String[] _FN = {"CarID", "Maker", "EMaker", "CarName", "ECarName",
                        "CarImageFile"};
        int[] _FT = {
            java.sql.Types.VARCHAR, java.sql.Types.VARCHAR,
            java.sql.Types.VARCHAR, java.sql.Types.VARCHAR,
            java.sql.Types.VARCHAR, java.sql.Types.VARCHAR
        };

       try {
            module = DataModuleFactory.getDataModule(fetchType);
            module.init(response.getOutputStream());
//          module.init(output);

            // set Parameter Info.
            module.addParameter("PARAM1", java.sql.Types.VARCHAR,
                                getEncode(request.getParameter("PARAM1")));
            module.addParameter("PARAM2", java.sql.Types.VARCHAR,
                                getEncode(request.getParameter("PARAM2")));

            // set SET Info.
            module.addSetInfo("SET_1", "", _FN, _FT);
        }catch(Exception ex){
            throw new ServletException(ex.getMessage());
        }

        try {
            module.startBinding();

            String query = "select * from car";

            module.startSet("SET_1");
            stmt = m_conn.createStatement();
            rs = stmt.executeQuery(query);
            while(rs.next()) {
                HashMap map = new HashMap();
                String a = rs.getString(_FN[0]);
                String b = rs.getString(_FN[1]);
                String c = rs.getString(_FN[2]);
                String d = rs.getString(_FN[3]);
                String e = rs.getString(_FN[4]);
                String f = rs.getString(_FN[5]);

                map.put(_FN[0], a);
                map.put(_FN[1], b);
                map.put(_FN[2], c);
                map.put(_FN[3], d);
                map.put(_FN[4], e);
                map.put(_FN[5], f);

                module.addRow("SET_1", map);
                System.out.print("select FN0="+_FN[0]+"  name="+a);
                System.out.print("    FN1="+_FN[1]+"  name="+b);
                System.out.print("    FN2="+_FN[2]+"  name="+c);
                System.out.print("    FN3="+_FN[3]+"  name="+d);
                System.out.print("    FN4="+_FN[4]+"  name="+e);
                System.out.print("    FN5="+_FN[5]+"  name="+f);
                System.out.println("");

            }
            module.endSet("SET_1");
            module.endBinding();

            rs.close(); rs = null;
            stmt.close(); stmt = null;

        }
        catch(Exception e) {
            e.printStackTrace();
            // 바인딩시 에러 전송은 sendBindErrorMessage메소드로 꼭 해야 한다.
            module.sendBindErrorMessage(e.toString());
        }
        finally {
            OutputStream out = response.getOutputStream();

            if(rs != null) {
                try { rs.close(); } catch(Exception e) {}
```

```
            }
            if(stmt != null) {
               try { stmt.close(); } catch(Exception e) {}
            }
         }
      }

      private void SET_Style(HttpServletRequest request, HttpServletResponse
response, String fetchType)
         throws ServletException, IOException
      {
         Statement stmt = null;
         ResultSet rs = null;

         DataModule module = null;
         try {
            // jasmin/JASMIN_SAMPLE2.odi
            // SET_1
            String[]   _FN   =   {"CarID",   "Maker",   "EMaker",   "CarName",
"ECarName", "CarImageFile" };
            int[]    _FT = {java.sql.Types.VARCHAR, java.sql.Types.VARCHAR,
                        java.sql.Types.VARCHAR, java.sql.Types.VARCHAR,
                        java.sql.Types.VARCHAR,java.sql.Types.VARCHAR};

            module = DataModuleFactory.getDataModule(fetchType);

            // set Parameter Info.
            module.addParameter("PARAM1",            java.sql.Types.VARCHAR,
getEncode(request.getParameter("PARAM1")));
            module.addParameter("PARAM2",            java.sql.Types.VARCHAR,
getEncode(request.getParameter("PARAM2")));

            String query = "select * from car";

            stmt = m_conn.createStatement();
            rs = stmt.executeQuery(query);

            module.makeSDM_SET("SET_1",rs,response.getOutputStream());

            rs.close(); rs = null;
            stmt.close(); stmt = null;

         }
         catch(Exception e) {
            e.printStackTrace();
            // set단위로 할 경우에 에러발생시 sendBindErrorMessage를 하지 않는다.
            // 왜냐하면 makeSDM_SET메소드 안에 처리가 되어 있다.
            throw new ServletException(e.getMessage());
         }
         finally {
            if(rs != null) {
               try { rs.close(); } catch(Exception e) {}
            }
            if(stmt != null) {
               try { stmt.close(); } catch(Exception e) {}
            }
         }

      }

      public void destroy()
      {
         super.destroy();
      }

      private String getEncode(String value) {
         try {
            return new String(value.getBytes("8859_1"), "KSC5601");
         }catch(Exception ex){
            return value;
         }
      }

}
```

Step 3 DataAction용 클래스 파일 만들기

DataAction을 실행하는 클래스 파일을 만듭니다. 본 예제에서는 아래와 같은 java 소스 파일을 이용하여 DataActionSampleServlet.class 파일로 만들었습니다.

```
package sample;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.*;

import java.util.*;

/**
 * <p>Title: OZ SDM API</p>
 * <p>Description: </p>
 * <p>Copyright: Copyright (c) 2005</p>
 * <p>Company: </p>
 * @author Forcs
 * @version 1.0
 */

public class DataActionSampleServlet extends HttpServlet
{
   private static final String _KEY_ODI_FETCH_TYPE = "_OZ_ODIFetchType_";
   private static final String _KEY_ODIITEM       = "_OZ_ODIITEM_";
   private static final String _KEY_ODICATEGORY    = "_OZ_ODICATEGORY_";
   private static final String _KEY_OZ_DATASET_    = "_OZ_DATASET_";

   private static final String _KEY_OZ_DAC_CNT = "_OZ_DAC_CNT";

   private static String TABLENAME = "car";

   private String dac_insert_query = "";
   private String dac_update_query = "";
   private String dac_delete_query = "";

   private PreparedStatement p_insert_stmt = null;
   private PreparedStatement p_update_stmt = null;
   private PreparedStatement p_delete_stmt = null;

   private Connection m_conn = null;

   public void init(ServletConfig config)
      throws ServletException
   {
      super.init(config);
   }

   public  void  doPost(HttpServletRequest  request,  HttpServletResponse
response)
      throws ServletException, IOException
   {
      process(request, response);
   }

   public  void  doGet(HttpServletRequest  request,  HttpServletResponse
response)
      throws ServletException, IOException
   {
      process(request, response);
   }

   private  void  process(HttpServletRequest  request,  HttpServletResponse
response)
      throws ServletException, IOException
   {

      try {
         String _URL = "jdbc:odbc:ozdemokr30";
         Properties prop = new Properties();
         prop.put("user", "");
         prop.put("password", "");
```

```
          Driver              driver              =              (Driver)
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver").newInstance();
          m_conn = driver.connect(_URL, prop);
       }
       catch(Exception ex) {
          response.getOutputStream().write(ex.getMessage().getBytes());
          throw new ServletException(ex.getMessage());
       }

       try {
          dac_insert_query = "";
          dac_update_query = "";
          dac_delete_query = "";

          Enumeration enum = request.getParameterNames();
          while (enum.hasMoreElements()) {
             String temp = (String) enum.nextElement();
             System.out.println("name=" + temp + "   value=" +
                              getEncode(request.getParameter(temp)));
          }

          //  ------------------------------------------------------------
---------
          // fetch parameters
          // "DM_BATCH_FETCH|DM_CONCURRENT_FETCH"
          String odiFetchType = getEncode(request.getParameter(
             _KEY_ODI_FETCH_TYPE));
          String item = getEncode(request.getParameter(_KEY_ODIITEM));
          String                          category                        =
getEncode(request.getParameter(_KEY_ODICATEGORY));
          String                           dataset                        =
getEncode(request.getParameter(_KEY_OZ_DATASET_));
          //  ------------------------------------------------------------
---------
          System.out.println("--------------------");
          if (odiFetchType == null) odiFetchType = "DM_CONCURRENT_FETCH";

          m_conn.setAutoCommit(false);

          //데이터Action의 카운트를 우선 계산한다.
          int                            dac_cnt                          =
Integer.parseInt(request.getParameter(_KEY_OZ_DAC_CNT));

          for (int i = 0; i < dac_cnt; i++) {
             String type = request.getParameter(i + ".TYPE");
             if (type.equalsIgnoreCase("insert")) {
                insert(request, response, i);
             }
             else if (type.equalsIgnoreCase("rowupdate")) {
                update(request, response, i);
             }
             else if (type.equalsIgnoreCase("delete")) {
                delete(request, response, i);
             }
             System.out.println(type+"
time=="+System.currentTimeMillis());
          }
          m_conn.commit();
          response.getOutputStream().write(new String("OK").getBytes());
       }catch(Exception ex){
          ex.printStackTrace();
          try { m_conn.rollback(); } catch(Exception e) {}
          response.getOutputStream().write(ex.getMessage().getBytes());
          throw new ServletException(ex.getMessage());

       }finally {
          if(p_insert_stmt != null) {
             try { p_insert_stmt.close(); } catch(Exception e) {}
          }
          if(p_update_stmt != null) {
             try { p_update_stmt.close(); } catch(Exception e) {}
          }
          if(p_delete_stmt != null) {
             try { p_delete_stmt.close(); } catch(Exception e) {}
          }

          if (m_conn != null) {
             try { m_conn.close(); } catch(Exception e) {}
```

```
            }

        }
    }

    private void insert(HttpServletRequest request, HttpServletResponse
response, int cnt)
        throws Exception
    {
        int                     insert_source_fieldCnt                    =
Integer.parseInt(request.getParameter(cnt + ".SRC_CNT"));

        if (dac_insert_query.equalsIgnoreCase("")) {
            //맨 처음 호출한 경우 쿼리문을 만들어서 prepared화 시킨다.
            dac_insert_query = "INSERT INTO "+TABLENAME+" (";
            for (int i = 0; i < insert_source_fieldCnt; i++) {
                String s_fieldName = getEncode(request.getParameter(cnt +
".SF_" + i));
                if (i == insert_source_fieldCnt - 1) {
                    dac_insert_query += s_fieldName;
                }
                else {
                    dac_insert_query += s_fieldName + ",";
                }
            }
            dac_insert_query += ") VALUES ( ";

            for (int i = 0; i < insert_source_fieldCnt; i++) {
                if (i == insert_source_fieldCnt - 1) {
                    dac_insert_query += "?";
                }
                else {
                    dac_insert_query += "? ,";
                }
            }

            dac_insert_query += ")";
            System.out.println("dac_insert_query="+dac_insert_query);
            p_insert_stmt = m_conn.prepareStatement(dac_insert_query);
        }

        for (int i = 0; i < insert_source_fieldCnt; i++) {
            String value = getEncode(request.getParameter(cnt + ".SV_" +
i));
            p_insert_stmt.setString(i + 1, value);
        }

        p_insert_stmt.execute();

    }

    private void update(HttpServletRequest request, HttpServletResponse
response, int cnt)
        throws Exception
    {

        int                     update_source_fieldCnt                    =
Integer.parseInt(request.getParameter(cnt + ".SRC_CNT"));
        int update_target_fieldCnt = 0;

        try {
            update_target_fieldCnt                                        =
Integer.parseInt(request.getParameter(cnt + ".TRG_CNT"));
        }catch(Exception ex){}

        if (dac_update_query.equalsIgnoreCase("")) {
            //맨 처음 호출한 경우 쿼리문을 만들어서 prepared화 시킨다.

            dac_update_query = "UPDATE "+TABLENAME+" set ";

            for (int i = 0; i < update_source_fieldCnt; i++) {
                String s_fieldName = getEncode(request.getParameter(cnt +
".SF_" + i));

                if (i == update_source_fieldCnt - 1) {
                    dac_update_query += s_fieldName +" = ? ";
                }
                else {
```

```
                dac_update_query += s_fieldName +" = ?, ";
             }
          }
          dac_update_query += " WHERE ";

          for (int i = 0; i < update_target_fieldCnt; i++) {
             String  t_fieldName  =  getEncode(request.getParameter(cnt  +
".DF_" + i));

             if (i == update_target_fieldCnt - 1) {
                dac_update_query += t_fieldName +" = ? ";
             }
             else {
                dac_update_query += t_fieldName +" = ? AND ";
             }
          }


          System.out.println("dac_update_query="+dac_update_query);
          p_update_stmt = m_conn.prepareStatement(dac_update_query);
       }

       int i = 0;
       for (i = 0; i < update_source_fieldCnt; i++) {
          String  value  =  getEncode(request.getParameter(cnt  +  ".SV_"  +
i));
          p_update_stmt.setString(i + 1, value);
       }

       for (int j = 0; j < update_source_fieldCnt; j++) {
          String  value  =  getEncode(request.getParameter(cnt  +  ".DV_"  +
j));
          p_update_stmt.setString(i + 1, value);
          i++;
       }
       p_update_stmt.execute();

   }

   private  void  delete(HttpServletRequest  request,  HttpServletResponse
response, int cnt)
         throws Exception
    {

        int                    delete_target_fieldCnt                    =
Integer.parseInt(request.getParameter(cnt + ".TRG_CNT"));

        if (dac_delete_query.equalsIgnoreCase("")) {
           //맨 처음 호출한 경우 쿼리문을 만들어서 prepared화 시킨다.

           dac_delete_query = "DELETE FROM "+TABLENAME+" WHERE ";

           for (int i = 0; i < delete_target_fieldCnt; i++) {
              String  t_fieldName  =  getEncode(request.getParameter(cnt  +
".DF_" + i));

              if (i == delete_target_fieldCnt - 1) {
                 dac_delete_query += t_fieldName +" = ? ";
              }
              else {
                 dac_delete_query += t_fieldName +" = ? AND ";
              }
           }

           System.out.println("dac_delete_query="+dac_delete_query);
           p_delete_stmt = m_conn.prepareStatement(dac_delete_query);
        }

        int i = 0;
        for (i = 0; i < delete_target_fieldCnt; i++) {
           String value  =  getEncode(request.getParameter(cnt  +  ".DV_"  +
i));
           p_delete_stmt.setString(i + 1, value);
        }

        p_delete_stmt.execute();

    }
```

```
    public void destroy()
    {
        super.destroy();
    }

    private String getEncode(String value) {
        try {
            return new String(value.getBytes("8859_1"), "KSC5601");
        }catch(Exception ex){
            return value;
        }
    }
}
```

Step 4 클래스 파일 등록하기

DataModuleSampleServlet.class, DataActionSampleServlet.class 파일을 웹 서버에 등록합니다.

- DataModuleSampleServlet.class, DataActionSampleServlet.class 파일을 웹 서버의 클래스 폴더에 복사합니다. 본 예제에서는 Tomcat 5.0의 webapps\ROOT\WEB-INF\classes 폴더에 sample 폴더를 새로 만든 후 두 개의 클래스 파일을 복사하였습니다.

- DataModuleSampleServlet.class, DataActionSampleServlet.class 파일을 웹 서버에 매핑합니다. 본 예제에서는 Tomcat 5.0의 webapps\ROOT\WEB-INF의 web.xml 파일을 편집기로 열어 아래와 같이 편집하여 복사한 클래스 파일을 매핑하였습니다.

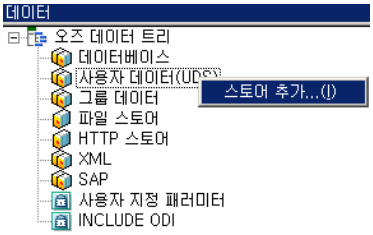
```
...

<!-- JSPC servlet mappings start -->
...
<servlet>
    <servlet-name>sample.DataModuleSampleServlet</servlet-name>
    <servlet-class>sample.DataModuleSampleServlet</servlet-class>
</servlet>

<servlet>
    <servlet-name>sample.DataActionSampleServlet</servlet-name>
    <servlet-class>sample.DataActionSampleServlet</servlet-class>
</servlet>

...
<servlet-mapping>
     <servlet-name>sample.DataModuleSampleServlet</servlet-name>
     <url-pattern>/sample.DataModuleSampleServlet</url-pattern>
</servlet-mapping>
```

```
<servlet-mapping>
     <servlet-name>sample.DataActionSampleServlet</servlet-name>
     <url-pattern>/sample.DataActionSampleServlet</url-pattern>
</servlet-mapping>

...
<!-- JSPC servlet mappings end -->
```

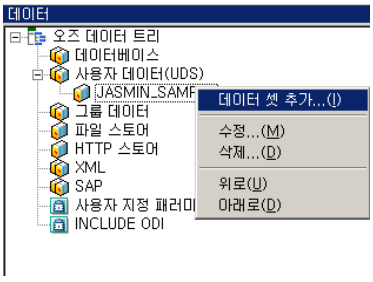
Step 5 ODI 만들기

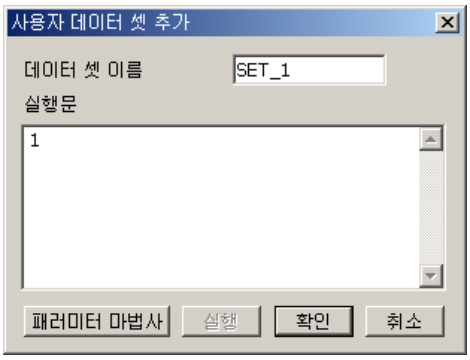
Servlet API를 이용할 경우에는 사용자 데이터(UDS) 스토어를 이용하고, 데이터셋에 사용자 정의 필드를 수동으로 추가하여야 합니다.

- 오즈 쿼리 디자이너를 실행하여 아래와 같이 사용자 데이터(UDS) 스토어를 추가합니다.

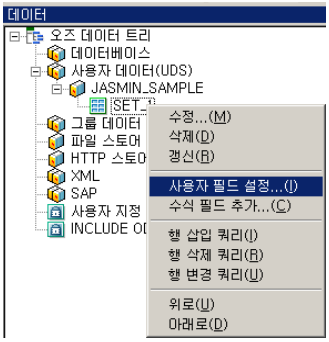


- 추가된 스토어에 데이터셋을 추가합니다.

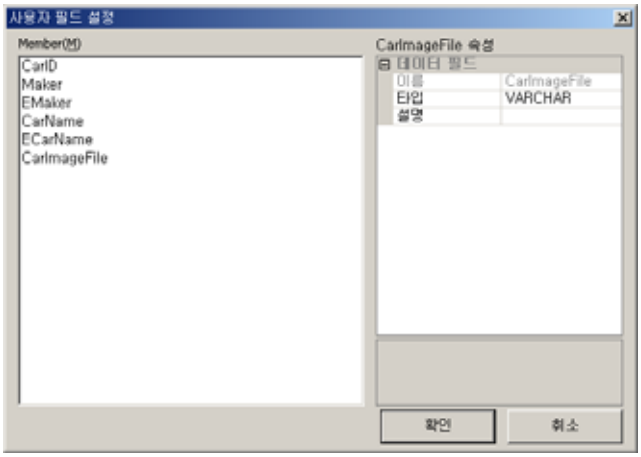




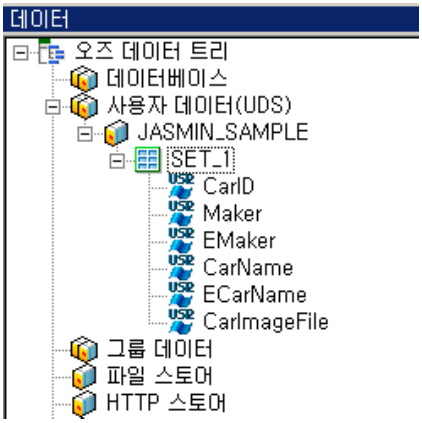
➢ 추가된 데이터셋에 사용자 필드를 추가합니다. 추가된 데이터셋을 마우스 오른쪽 버튼으로 클릭하여 나타나는 팝업 메뉴에서 [사용자 필드 설정] 메뉴를 클릭합니다.



➢ 사용자 필드 설정 다이얼로그에서 아래 그림처럼 사용자 필드를 추가한 후 [확인] 버튼을 누릅니다.



데이터 정보 트리에 사용자 필드가 아래 그림처럼 추가되었는지 확인합니다.



➢ SET_1을 마우스 오른쪽 버튼으로 클릭하여 각각 [행 삽입 쿼리], [행 삭제 쿼리], [행 변경 쿼리] 메뉴를 눌러 각각 행 삽입 쿼리, 행 삭제 쿼리, 행 변경 쿼리문을 입력합니다.

입력한 행 삽입 쿼리문은 다음과 같습니다.

```
INSERT INTO car(#@ARG_SF1#, #@ARG_SF2#, #@ARG_SF3#, #@ARG_SF4#,
#@ARG_SF5#, #@ARG_SF6#)
VALUES('#@ARG_SV1#','#@ARG_SV2#', '#@ARG_SV3#','#@ARG_SV4#',
'#@ARG_SV5#', '#@ARG_SV6#')
```

입력한 행 삭제 쿼리문은 다음과 같습니다.

```
DELETE FROM car WHERE [#@ARG_DF1#] = '#@ARG_DV1#";
```

입력한 행 변경 쿼리문은 다음과 같습니다.

```
UPDATE car SET
[#@ARG_SF1#] = '#@ARG_SV1#', [#@ARG_SF2#] = '#@ARG_SV2#',
[#@ARG_SF3#] = '#@ARG_SV3#', [#@ARG_SF4#] = '#@ARG_SV4#',
[#@ARG_SF5#] = '#@ARG_SV5#', [#@ARG_SF6#] = '#@ARG_SV6#'
WHERE [#@ARG_DF1#] = '#@ARG_DV1#'
```

➢ ODI 파일을 "JASMIN_SAMPLE4.odi" 파일로 저장합니다.

Step 6 OZF와 프로젝트 만들기

※ 주의사항 : Servlet API를 이용하여 데이터 연동을 오즈 서버가 아닌 다른 서버와 연동할 경우 연동할 서버의 URL을 지정하기 위해서는 DataModule의 RegisterUserDataModule 함수를 이용하여 JavaScript 객체를 등록하여야 합니다. JavaScript 객체를 등록하는 함수는 반드시 전역 함수로 작성되어야 하므로, Document.GlobalFunction에 작성하거나 OZF로 작성한 후 해당 OZF 파일을 프로젝트에 추가하여 사용합니다. 본 예제

에서는 OZF로 작성한 후 프로젝트에 추가하는 예로 설명하겠습니다.

➢ 오즈 애플리케이션 함수 편집기를 실행하여 OZF를 새로 만든 후 스크립트 편집창에 아래와 같이 입력한 후 "MyFrameworkURLUDS.ozf"로 저장합니다.

```
MyFrameworkURLUDS.prototype.GetFrameworkURL =
  MyFrameworkURLUDS_GetFrameworkURL;
MyFrameworkURLUDS.prototype.GetCUDFrameworkURL =
  MyFrameworkURLUDS_GetCUDFrameworkURL;
MyFrameworkURLUDS.prototype.GetFrameworkPostParam =
  MyFrameworkURLUDS_GetFrameworkPostParam;
MyFrameworkURLUDS.prototype.GetCUDFrameworkPostParam =
  MyFrameworkURLUDS_GetCUDFrameworkPostParam;

function      MyFrameworkURLUDS(_url,      _cud_url,      _url_param,
_cud_url_param){
    this.url = _url;
    this.cud_url = _cud_url;
    this.url_param = _url_param;
    if(this.url_param == null){
          this.url_param = "";
     }
    this.cud_url_param = _cud_url_param;
    if(this.cud_url_param == null){
          this.cud_url_param = "";
     }
}

function MyFrameworkURLUDS_GetFrameworkURL(dataset_name){
    return this.url;
}

function MyFrameworkURLUDS_GetCUDFrameworkURL(dataset_name){
    return this.cud_url;
}

function MyFrameworkURLUDS_GetFrameworkPostParam(dataset_name){
    return "default&"+this.url_param;
}

function MyFrameworkURLUDS_GetCUDFrameworkPostParam(dataset_name){
    return "default&"+this.cud_url_param;
}
```

※ 참고사항

① 객체에는 GetFrameworkURL과 GetCUDFrameworkURL 함수가 반드시 입력되어 있어야 합니다.

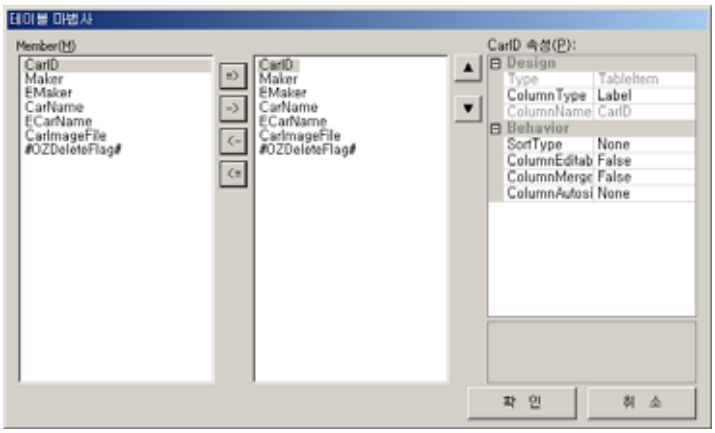
② GetFrameworkURL 함수는 서버로부터 데이터를 가져올 경우 사용되는 URL을 얻을 때 호출됩니다.

③ GetCUDFrameworkURL 함수는 서버에 DataAction을 보내는 경우 사용되는 URL을 얻을 때 호출됩니다.

④ 위 예제에서는 데이터셋 이름과 상관없이 항상 같은 URL을 사용하도록 구현한 예이며, 데이터셋 이름에 따라 URL을 다르게 사용할 수도 있습니다.

➢ 오즈 애플리케이션 디자이너를 실행하여 프로젝트를 새로 만든 후 프로젝트에 "JASMIN_SAMPLE4.odi" 파일과 "MyFrameworkURLUDS.ozf" 파일을 추가합니다.

➢ Board에 Table을 추가한 후 Table의 ODIKey 속성값을 "JASMIN_SAMPLE4"로, DataSet 속성값을 "SET_1"을 설정한 후 테이블 마법사를 실행하여, 아래 그림처럼 테이블에 표시할 필드를 설정합니다.



➢ Table에서 데이터를 삽입, 삭제, 변경하기 위해서 Table의 속성 중 "AllowInsert", "AllowDelete", "AllowUpdate" 속성값을 모두 "True"로 변경합니다.

➢ Table의 OnInitialize 이벤트에 아래와 같이 스크립트를 입력합니다.

```
var uds = new MyFrameworkURLUDS("http://127.0.0.1:8088/sample.DataModuleSampleServlet","http://127.0.0.1:8088/sample.DataActionSampleServlet");
```

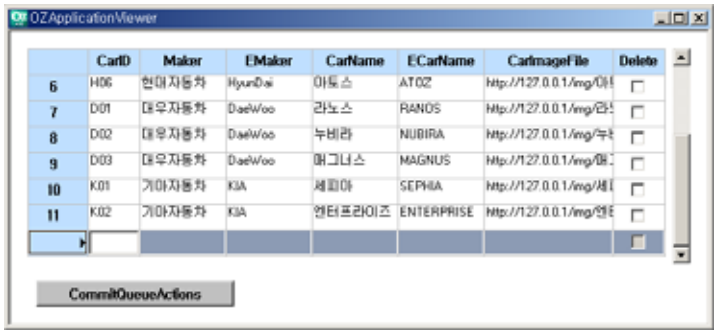
```
var datamanager = _GetDataManager();
var datamodule = datamanager.GetDataModule("JASMIN_SAMPLE4");
datamodule.RegisterUserDataModule(uds);
```

➢ Board에 Button을 추가한 후 Button의 OnClick 이벤트에 아래와 같이 스크립트를 입력합니다.

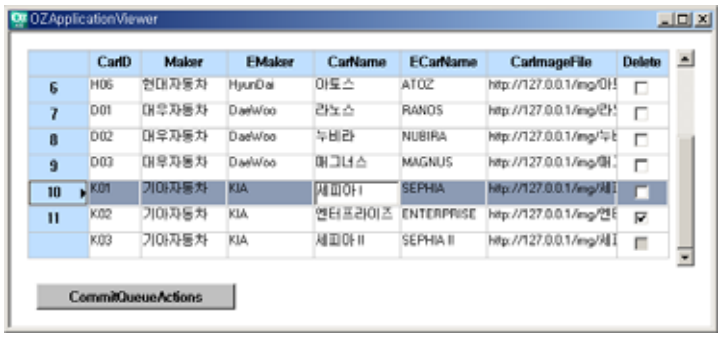
```
var result = Table1.CommitQueuedActions();
if(result == "") {
   Table1.GetDataModule().RefreshAllDataSet();
} else {
   _MessageBox(result);
}

//var myODIObject =
  _GetDataManager().GetDataModule("odiName").GetUserDataModule();
//Select할때 기본 패러미터에 다른 패러미터를 추가하고자 할 때 사용
//  myODIObject.url_param = "key1=value1&key2=value2";
//DataAction할때 기본 패러미터에 다른 패러미터를 추가하고자 할 때 사용
//  myODIObject.cud_url_param = "key1=value1&key2=value2";
//_GetDataManager().GetDataModule("odiName").RefreshAllDataSet();
```

Step 7 미리보기

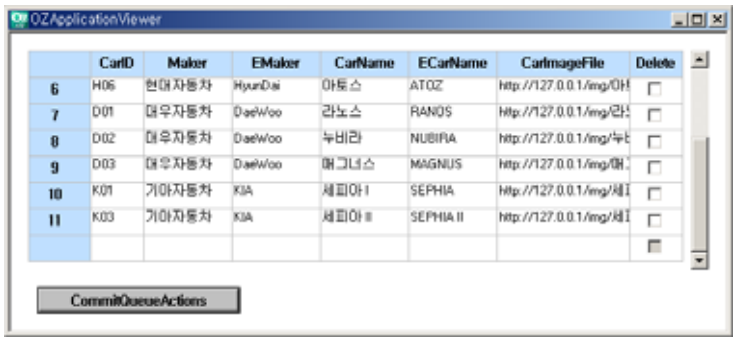
웹 서버를 구동한 후 오즈 애플리케이션 디자이너 툴바에서 미리보기 아이콘을 클릭하여 미리보기 합니다.



맨 마지막에 비어있는 행에 추가할 데이터 내용을 입력합니다. 삭제할 행은 Delete 컬럼의 체크 박스를 체크합니다. 변경할 데이터 내용을 직접 입력하여 변경합니다.



입력, 삭제, 변경한대로 DataAction을 실행하려면 버튼을 클릭하여 적용합니다. [CommitQueueActions] 버튼을 클릭하여 적용한 모습은 아래 그림과 같습니다.



Servlet API 예 2 : Report - 데이터 검색

다음으로 Servlet API를 이용하여 테이블에 데이터를 표시하는 보고서 파일을 웹 브라우저에서 호출하는 예제를 살펴보겠습니다. Servlet API를 연동할 때에는 뷰어 패러미터 중 framworkurl 패러미터를 이용하여 클래스 파일의 경로(HTTP URL)를 설정하여야 합니다. 관련된 뷰어 패러미터는 odi.odi명.framworkurl, odi.framworkurl, connection.framworkurl 입니다.

※ 주의사항 : Servlet API를 이용하여 보고서에 데이터를 표시할 경우에는 오즈 리포트 디자이너에서는 미리보기할 수 없으며, 서버 리파지토리에 보고서를 등록한 후 웹 페이지를 통해서만 미리보기할 수 있습니다.

Step 1 ozsdmapi.jar 파일을 웹 서버에 등록하기

Servlet API 연동 모듈인 ozsdmapi.jar 파일을 웹 서버의 라이브러리 폴더에 복사하여 등록하였습니다. 본 예제에서는 Tomcat 5.0의 webapps\ROOT\WEB-INF\lib 폴더에 ozsdmapi.jar 파일을 복사하였습니다.

Step 2 클래스 파일 등록하기

데이터 검색용 클래스 파일인 DataModuleSampleServlet.class 파일을 웹 서버에 등록합니다.

- DataModuleSampleServlet.class 파일을 웹 서버의 클래스 폴더에 복사합니다. 본 예제에서는 Tomcat 5.0의 webapps\ROOT\WEB-INF\classes\sample 폴더에 클래스 파일을 복사하였습니다.

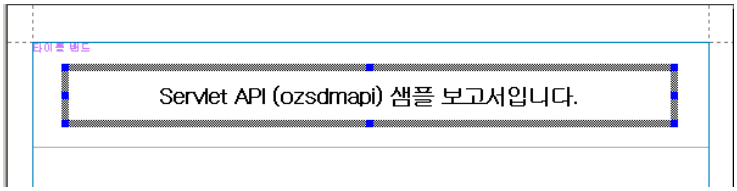
- DataModuleSampleServlet.class 파일을 웹 서버에 매핑합니다. 본 예제에서는 Tomcat 5.0의 webapps\ROOT\WEB-INF의 web.xml 파일을 편집기로 열어 아래와 같이 편집하여 복사한 클래스 파일을 매핑하였습니다.

```
...

<!-- JSPC servlet mappings start -->
...
<servlet>
   <servlet-name>sample.DataModuleSampleServlet</servlet-name>
   <servlet-class>sample.DataModuleSampleServlet</servlet-class>
</servlet>
...
<servlet-mapping>
     <servlet-name>sample.DataModuleSampleServlet</servlet-name>
     <url-pattern>/sample.DataModuleSampleServlet</url-pattern>
</servlet-mapping>
...
<!-- JSPC servlet mappings end -->
```

Step 3 보고서 만들기

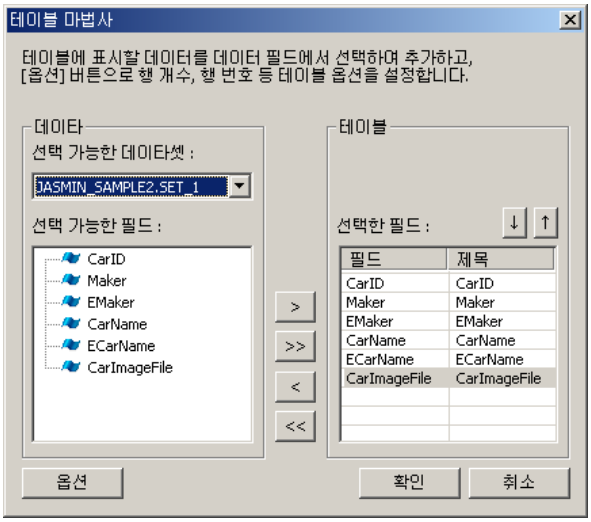
- 오즈 리포트 디자이너를 실행하여 보고서를 새로 만든 후 JASMIN_SAMPLE2.odi 파일을 추가합니다.

- 보고서에 타이틀 밴드를 추가하고, 타이틀 밴드에 라벨을 추가하여 보고서 타이틀을 만듭니다.

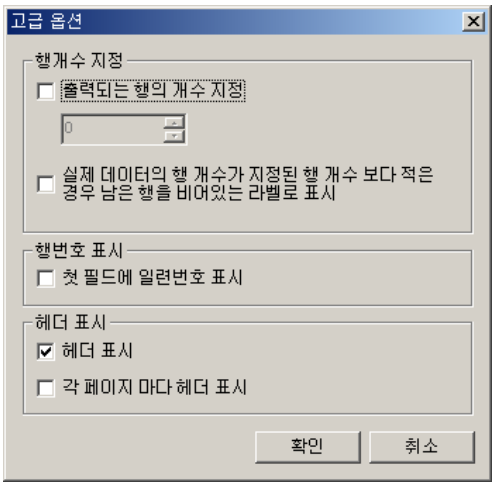


- 보고서에 데이터 밴드를 추가한 후 데이터 밴드의 속성 중 "ODI 이름" 속성값을 "JASMIN_SAMPLE2"로, "데이터셋" 속성값을 "SET_1"으로 설정합니다.

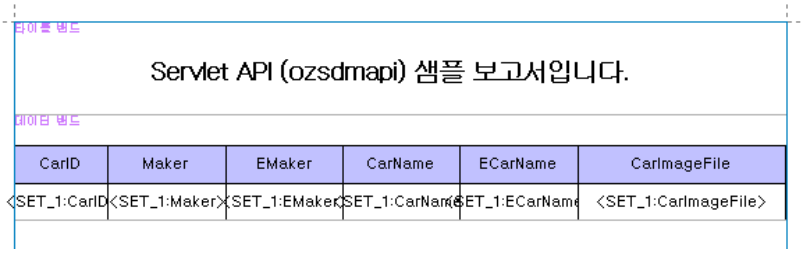
- 툴바에서 테이블 아이콘을 선택한 후 데이터 밴드에 원하는 테이블 크기만큼 드래그&드롭하면 자동으로 테이블 마법사가 실행됩니다. 테이블 마법사에서 테이블에 표시할 필드를 추가합니다.



- 테이블의 헤더를 표시하기 위해 테이블 마법사의 [옵션] 버튼을 클릭하여 나타나는 테이블 옵션 다이얼로그에서 "헤더 표시" 옵션을 체크한 후 [확인] 버튼을 클릭하여 옵션을 적용합니다.



- 옵션 다이얼로그가 닫힌 후에 테이블 마법사의 [확인] 버튼을 클릭하여 데이터 밴드에 테이블을 추가한 후 각 셀의 너비, 글꼴, 배경색 등을 디자인합니다.



➢ 디자인된 보고서를 "Sample_ServletAPI.ozr" 파일로 저장합니다.

Step 4 보고서 미리보기

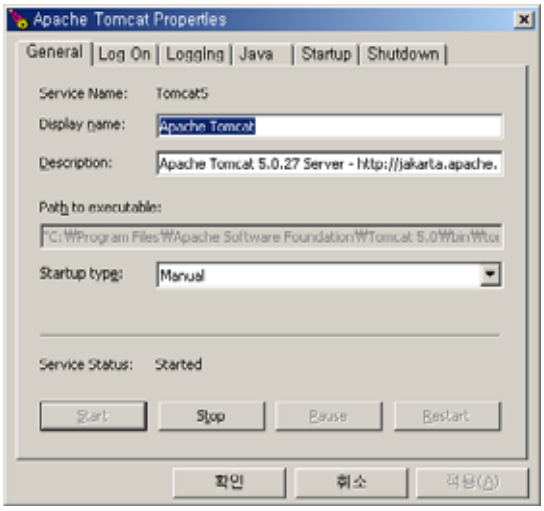
➢ 작성된 보고서를 웹에서 호출하는 웹 페이지를 작성합니다. 본 예제에서는 아래와 같이 입력한 후 웹 서버의 루트 경로에 Sample_ServletAPI.htm 파일로 저장하였습니다.

```
<HTML>
<BODY>
 <OBJECT width = "0" height = "0" ID="ZTransferX"
CLASSID="CLSID:C7C7225A-9476-47AC-B0B0-FF3B79D55E67"
codebase="127.0.0.1:8088/ozrviewer/ZTransferX.cab#version=2,0,1,2">
   <PARAM NAME="download.Server" VALUE="http://127.0.0.1/ozrviewer">
   <PARAM NAME="download.Port" VALUE="8088">
   <PARAM NAME="download.Instruction" VALUE="ozrviewer.idf">
   <PARAM NAME="install.Base" VALUE="<PROGRAMS>/Forcs">
   <PARAM NAME="install.Namespace" VALUE="Sample_ServletAPI">
 </OBJECT>
<OBJECT id = "ozrviewer" CLASSID="CLSID:0DEF32F8-170F-46f8-B1FF-
4BF7443F5F25" width="100%" height="100%">
   <param name="connection.servlet"
value="http://127.0.0.1:8088/OZServlet35/server">
   <param name="connection.reportname" value="Sample_ServletAPI.ozr">
   <param name="odi.odinames" value="JASMIN_SAMPLE2">
   <param name="odi.JASMIN_SAMPLE2.frameworkurl"
value="http://127.0.0.1:8088/sample.DataModuleSampleServlet">
   <param name="odi.JASMIN_SAMPLE2.fetchtype" value="BATCH">
   <param name="viewer.isframe" value="false">
   <param name="viewer.namespace" value="Sample_ServletAPI\ozviewer">
 </OBJECT>
</body>
</HTML>
```

※ 참고사항 : Servlet API를 이용하여 데이터를 바인딩할 경우에는 뷰어

패러미터 중 odi.odi명.framworkurl 또는 odi.framworkurl 또는 connection.framworkurl 패러미터 값에 연동할 클래스 파일에 대한 URL 경로를 반드시 설정하여야 합니다.

➢ 작성된 보고서를 오즈 서블릿 서버의 리파지토리 경로에 복사한 후 웹 서버와 오즈 서블릿 서버를 구동시킵니다.



➢ 웹 브라우저를 실행하여 보고서 파일을 호출하는 웹 페이지를 실행하여 보고서를 미리보기합니다.