목 차

Ⅰ. 오즈 엔터프라이즈 닷넷 서버 API

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

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

다음은 오즈 엔터프라이즈 닷넷 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 ADO .NET/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.

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

라이브러리 파일명	설명
OZServer.NET.dll	오즈 닷넷 서버와 접속하기 위한 파일입니다.
SAP.Connector.dll	SAP과 접속하기 위한 파일입니다.
Interop.MSScriptControl.dll	Jscript를 실행하기 뒤한 COM 모듈입니다.

Class Cache

Constructor Summary

- **Cache(string url, string id, string pw, boolean autoLogin, boolean useUSL)**

Method Summary

- **OZAttributeList GetConfiguration()**
- **void SetConfiguration(OZAttributeList attrs)**

Constructor Detail

ASP .NET	`public Cache(string url, string id, string pw, bool autoLogin, bool useUSL)`	
Argument	*url*	오즈 엔터프라이즈 닷넷 서버의 URL ex) string url = "http://127.0.0.1/oz/server.aspx";
	id	사용자 아이디 ex) string id = "admin";
	pw	사용자 패스워드 ex) string pw = "admin";
	autoLogin	자동 로그인 여부 ex) boolean autoLogin = true;
	useUSL	USL 사용 여부 ex) boolean useUSL = false;

Method Detail

- **getCacheConfigration**

Prototype	`public OZAttributeList GetCacheConfigration () throws OZAPIException`

Definition	오즈 서버의 캐시 설정을 가져옵니다. 조회 내용은 캐시 키 사용여부, 캐시 저장 경로, 메모리 캐시 타임 아웃, 디스크 캐시 타임 아웃, 프리 메모리 퍼센트입니다.

■ **setCacheConfiguration**

Prototype	`public void SetCacheConfiguration (OZAttributeList attrs) throws OZAPIException`
Definition	캐시 설정을 변경합니다. 캐시 설정 변경은 해당 변경된 값을 캐시 키 사용여부, 메모리 캐시 타임 아웃, 디스크 캐시 타임 아웃, 프리 메모리 퍼센트, 캐시 저장 경로명 등의 요소로 변경합니다
Argument	*attrs* 캐시 설정 속성값

관련 Class

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

API를 사용할 때 생성되는 Exception입니다. 모든 API에서 OZAPIException을 발생시킵니다.

- 메소드
  - getMessage

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

■ **OZAttributeList (oz.util.OZAttributeList.cs) of GetCacheConfigration, SetCacheConfiguration**

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

- 메소드
  - This[string key] {get; set;}

Prototype	This[string key] {get; set;}
Definition	해당 key에 해당하는 속성값을 가져옵니다.

➢ Key

Property를 설정하거나 얻어오는 경우 사용되는 key는 다음 표와 같습니다.

Key	Value	설명
Active	"true" "false"	캐시키 사용 여부 선택 ex) p["datamodule.active"] = "false";
CACHE_FILE_PATH	저장 경로 (string)	캐시 저장 경로명 설정 ex) p["CACHE_FILE_PATH"] = "%OZ_HOME%/cache";
DM_CACHE_FILE_PATH	저장 경로 (string)	Data Module 캐시 저장 경로 ex) p["DM_CACHE_FILE_PATH"] = "%OZ_HOME%/cache_dm/";
memoryCacheValidTime	시간 (Unit second)	메모리 캐시 타임 아웃(단위:초) ex) p["datamodule.memoryCacheValidTime"] = "100";
diskCacheValidTime	시간 (Unit second)	디스크 캐시 타임 아웃(단위:초) ex) p["datamodule.diskCacheValidTime"] = "100";
FreeMemoryPercentage	프리메모리 (%)	프리 메모리 퍼센트 설정 ex) p["datamodule.freeMemoryPercentage"] = "20";

※ 참고사항 : 캐시 키의 설정 방법은 "오즈 엔터프라이즈 서버 관리자 매뉴얼의 오즈 엔터프라이즈 서버 설정-cachemngr.properties"를 참조하시기 바랍니다.

Sample : CacheSample.cs

```
using System;

using oz.framework.api;

namespace sample{
/// <summary>
/// Cache
/// </summary>
   public class CacheTest{
      public static void Main(){
         string url = "http://127.0.0.1/oztest/server.aspx";
         string id = "admin";
          string password = "admin";
          Cache c = new Cache(url, id, password, true, true);
          oz.util.OZAttributeList attrs = c.GetConfiguration();
          Console.WriteLine(attrs);
          c.SetConfiguration(attrs);
```

```
        }
    }
}
```

Class ConnectionPool

Constructor Summary

- **ConnectionPool(string url, string id, string pw, boolean autoLogin, boolean useUSL)**

Method Summary

- **void AddPool(ConnectionPoolInfo pool)**
- **void RemovePool(string alias)**
- **ConnectionPoolInfo[] GetPoolInfos()**
- **ConnectionPoolStatus[] GetPoolStatuses()**
- **ConnectionPoolInfo GetPoolInfo(string alias)**
- **void Save()**

Constructor Detail

ASP .NET	`public ConnectionPool(string url, string id, string pw, boolean autoLogin, boolean useUSL)`	
Argument	*url*	오즈 닷넷 서버의 URL ex) string url = "http://127.0.0.1/oz/server.aspx";
	id	사용자 아이디 ex) string id = "admin";
	pw	사용자 패스워드 ex) string pw = "admin";
	autoLogin	자동 로그인 여부 ex) boolean autoLogin = true;
	useUSL	USL 사용 여부 ex) boolean useUSL = false;

Method Detail

AddPool

Prototype	`public void AddPool(ConnectionPoolInfo pool) throws OZAPIException`
Definition	커넥션 풀 인포에 들어있는 정보에 맞춰서 커넥션 풀을 만들어 추가합니다. 커넥션 풀 인포에는 상태, 이름, 타입, 주소, 포트, SID, DB드라이버 종류, DB 서버이름, 사용자명, 암호, 최대접속, 초기 접속, 타임 아웃, URL, 프로퍼티, 사용자정의 드라이버가 있습니다.
Argument	*pool* 추가할 ConnectionPoolInfo의 내용

RemovePool

Prototype	`public void RemovePool(string alias) throws OZAPIException`
Definition	해당 이름의 ConnectionPool을 삭제합니다.
Argument	*alias* ConnectionPool 이름

GetPoolInfos

Prototype	`public ConnectionPoolInfo[] GetPoolInfos() throws OZAPIException`
Definition	모든 ConnectionPool에 대한 정보를 가져옵니다.

GetPoolStatuses

Prototype	`public ConnectionPooStatus[] GetPoolStatuses() throws OZAPIException`
Definition	모든 ConnectionPool에 대한 상태 정보를 가져옵니다.

GetPoolInfo

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

Save

Prototype	`public void Save() throws OZAPIException throws OZAPIException`
Definition	서버에 모든 ConnectionPool의 정보를 저장합니다.

관련 Class

- **ConnectionPoolInfo(oz.framework.db.ConnectionPoolInfo)**

  데이터베이스 연결에 대한 각종 정보들을 가지고 있는 클래스입니다.

- **ConnectionPoolStatus(oz.framework.db.ConnectionPoolStatus)**

  ConnectionPool의 상태 정보를 멤버 변수로 가지고 있는 클래스입니다.

  - 멤버 변수

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

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

Sample : ConnectionPoolTest.cs

```
using System;

using oz.framework.api;
using oz.framework.db;

namespace sample
{
	/// <summary>
	/// ConnectionPoolTest
	/// </summary>
	public class ConnectionPoolTest
	{
		public static void Main()
		{
	   string url = "http://127.0.0.1/oztest/server.aspx";
	   string id = "admin";
	   string password = "admin";

	   ConnectionPool cp = new ConnectionPool(url, id, password,
true, true);

	   string alias = "connection_pool_test";
```

```
            ConnectionPoolInfo poolInfo = new ConnectionPoolInfo();
            poolInfo.Alias = alias;
            poolInfo.Vendor = "MSSQL";
            poolInfo.Items["serverAddress"] = "218.36.12.88";
            poolInfo.Items["portNo"] = "1433";
            poolInfo.Items["dbName"] = "QATEST";
            poolInfo.Items["user"] = "user1";
            poolInfo.Items["password"] = "user123";

            poolInfo.MaxConnections = 20;
            poolInfo.InitialConnections = 5;
            poolInfo.TimeOut = 5;
            cp.AddPool(poolInfo);

            ConnectionPoolInfo addedPoolInfo = cp.GetPoolInfo(alias);
            Console.WriteLine(addedPoolInfo);

            cp.RemovePool(alias);

            ConnectionPoolInfo[] poolInfos = cp.GetPoolInfos();
            foreach(ConnectionPoolInfo cpi in poolInfos)
            {
                Console.WriteLine(cpi);
            }

            ConnectionPoolStatus[] statuses = cp.GetPoolStatuses();
            foreach(ConnectionPoolStatus status in statuses)
            {
                Console.WriteLine(status);
            }
                 }
        }
}
```

Class DataBind

Constructor Summary

- **DataBind(string url, string id, string pw, boolean autoLogin, boolean useUSL)**

Method Summary

- **void SetConfigration(OZAttributeList config)**
- **OZAttributeList GetConfigration()**

Constructor Detail

ASP .NET	`public DataBind(string url, string id, string pw, boolean autoLogin, boolean useUSL)`	
Argument	*url*	오즈 닷넷 서버의 URL ex) string url = "http://127.0.0.1/oz/server.aspx";
	id	사용자 아이디 ex) string id = "admin";
	pw	사용자 패스워드 ex) string pw = "admin";
	autoLogin	자동 로그인 여부 ex) boolean autoLogin = true;
	useUSL	USL 사용 여부 ex) boolean useUSL = false;

Method Detail

- **SetConfigration**

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

■ **GetConfigration**

Prototype	`public OZAttributeList GetConfigration() throws OZAPIException`
Definition	DataBind의 설정값 즉, "databind.properties"에 설정되어 있는 값을 가져옵니다.

- Key

SetConfigration()와 GetConfigration()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 : DataBindTest.cs

```
using System;

using oz.util;
using oz.framework.api;

namespace sample
```

```
{
       /// <summary>
       /// DataBindTest
       /// </summary>
       public class DataBindTest
       {
       public static void Main()
       {
          string url = "http://127.0.0.1/oztest/server.aspx";
          string id = "admin";
          string password = "admin";

          DataBind db = new DataBind(url, id, password, true, true);

          OZAttributeList attrs = db.GetConfiguration();
          foreach(StringDictionaryEntry attr in attrs)
          {
             Console.WriteLine(attr.Key + " : " + attr.Value);
          }

          db.SetConfiguration(attrs);
              }
       }
}
```

Class Log

Constructor Summary

- **Log(string url, string id, string pw, boolean autoLogin, boolean useUSL)**

Method Summary

- **string GetConfiguration ()**
- **Stream DownloadLog ()**
- **Stream DownloadLog(string fileName)**
- **void SetConfiguration (string config)**
- **string[] GetFileNames()**

Constructor Detail

ASP .NET	`public Log(string url, string id, string pw, boolean autoLogin, boolean useUSL)`	
Argument	*url*	오즈 닷넷 서버의 URL ex) string url = "http://127.0.0.1/oz/server.aspx";
	id	사용자 아이디 ex) string id = "admin";
	pw	사용자 패스워드 ex) string pw = "admin";
	autoLogin	자동 로그인 여부 ex) boolean autoLogin = true;
	useUSL	USL 사용 여부 ex) boolean useUSL = false;

Method Detail

GetConfiguration

Prototype	`public string GetConfiguration() throws OZAPIException`
Definition	로그 설정을 서버로부터 가져옵니다.

DownloadLog

Prototype	`public stream DownloadLog() throws OZAPIException`
Definition	서버로부터 로그 파일을 다운로드합니다.

DownloadLog

Prototype	`public stream DownloadLog(string fileName) throws OZAPIException`	
Definition	서버로부터 다운로드한 로그 파일을 로컬에 저장합니다.	
Argument	*`fileName`*	저장할 파일명

SetConfiguration

Prototype	`public void SetConfiguration(string logs) throws OZAPIException`	
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

GetFileNames

Prototype	`public string[] GetFileNames() throws OZAPIException`
Definition	생성된 로그 파일명을 가져옵니다.

Sample : LogSample.cs

```
using System;
using System.IO;

using oz.framework.api;
```

```
namespace sample{
/// <summary>
/// LogTest
/// </summary>
   public class LogTest{
      public static void Main(){
       string url = "http://127.0.0.1/oz/server.aspx";
       string id = "admin";
       string password = "admin";

       Log log = new Log(url, id, password, true, true);

       string config = log.GetConfiguration();
       Console.WriteLine(config);

       string[] logs = log.GetFileNames();
       foreach(string s in logs){
          Console.WriteLine(s);
       }
       Stream logFile = log.DownloadLog();
       }
   }
}
```

Class Module

Constructor Summary

- **Module(string url, string id, string pw, boolean autoLogin, boolean useUSL)**

Method Summary

- **Stream GetOZD(string item, string category, string[] urls)**
- **stream GetOZU(string item, string category, string[] urls)**
- **void AddODIParameter(string odiName, string key, string value)**
- **void AddODIParameter(string odiName, string item, string category, , IDictionary parameters)**
- **void AddParameter(string key, string value)**
- **void AddApplicationParameter(string key, string value)**
- **void SaveOZD(string fileName, string item, string category, string[] urls)**
- **void SaveOZU(string fileName, string item, string category, string[] urls)**
- **public IReportInfo AddReport(string itemName, string categoryName)**
- **public IReportInfo AddReport(string itemName, string categoryName, String displayName)**
- **public Stream GetOZD()**
- **public void SaveOZD(string fileName)**

Constructor Detail

ASP .NET	`public Module(string url, string id, string pw, boolean autoLogin, boolean useUSL)`	
Argument	*url*	오즈 닷넷 서버의 URL ex) string url = "http://127.0.0.1/oz/server.aspx";
	id	사용자 아이디 ex) string id = "admin";

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

Method Detail

■ getOZD

Prototype	`public stream GetOZD(string item, string category, string[] urls) throws OZAPIException`	
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="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
	urls	OZD에 함께 저장할 이미지 파일의 URL

■ GetOZU

Prototype	`public Stream getOZU(String item, String category, String[] urls) throws OZAPIException`	
Definition	서버로부터 애플리케이션 파일과 SDM 파일을 가져와 OZU 파일을 생성하여 가져옵니다. ※ 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. ※ 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) throws OZAPIException	
Definition	SDM 파일을 만들기 위해 ODI에 지정되어 있는 ODI 패러미터 값을 설정합니다. ODI 패러미터 값을 설정하지 않을 경우에는 ODI 기본 패러미터의 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, IDictionary parameters) throws OZAPIException	
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
	parameters	패러미터 Key, Value 가 설정되어 있는 Dictionary

※ 주의사항 : OZU 생성시 parameters 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) throws OZAPIException

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) throws OZAPIException`	
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

■ **SaveOZD**

Prototype	`public void SaveOZD(string fileName, string item, string category, string[] urls) throws OZAPIException`	
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 OZAPIException`	
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

■ **AddReport**

Prototype	`public IReportInfo AddReport(string itemName, string categoryName)`
Definition	다중 보고서를 하나의 OZD 파일로 만들기 위하여 리포트를 추가합니다.
Argument	*itemName* 리포트를 추가할 아이템 이름
	categoryName 리포트를 추가할 아이템의 카테고리 이름

Prototype	`public IReportInfo AddReport(string itemName, string categoryName, String displayName)`
Definition	다중 보고서를 하나의 OZD 파일로 만들기 위하여 리포트를 추가합니다.
Argument	*itemName* 리포트를 추가할 아이템 이름
	categoryName 리포트를 추가할 아이템의 카테고리 이름
	displayname 뷰어의 보고서 트리에 표시할 보고서 이름

■ **GetOZD**

Prototype	`public Stream GetOZD()`
Definition	다중 보고서를 갖는 OZD를 생성한 후 입력 스트림을 가져옵니다.

■ **SaveOZD**

Prototype	`public void SaveOZD(string fileName)`
Definition	다중 보고서를 갖는 OZD를 생성한 후 특정 경로에 OZD 파일을 저장합니다.
Argument	*filename* 저장할 카테고리 경로 및 파일 이름

관련 Interface

■ **IReportInfo(oz.framework.api.IReportInfo)**

다중 보고서를 갖는 OZD 생성 시 보고서에 대한 부가 정보를 설정합니다.

- **ItemName**

Prototype	`string ItemName{get;}`
Definition	아이템 이름을 가져옵니다.

- **CategoryName**

Prototype	`string CategoryName{get;}`
Definition	아이템의 카테고리 이름을 가져옵니다.

- **AddURL**

Prototype	`void AddURL(params string[] urls)`
Definition	보고서에 포함된 이미지 파일의 URL을 배열로 설정합니다.
Argument	*urls* 이미지 파일 URL

- **AddParameter**

Prototype	`void AddParameter(string key, string value)`	
Definition	패러미터 값을 설정합니다.	
Argument	*key*	패러미터 이름
	value	패러미터 값

- **AddODIParameter**

Prototype	`public void AddODIParameter(string odiName, string key, string value)`	
Definition	ODI 패러미터 값을 설정합니다.	
Argument	*odiName*	ODI 이름
	key	ODI 패러미터 이름
	value	ODI 패러미터 값

Sample : ModuleSample.cs

```
using System;
using System.IO;
using System.Collections;

using oz.framework.api;

namespace sample{
/// <summary>
///
```

```
/// ModuleTest - Before start
/// You need to customize parameters to run in your environment
/// We don't provide oza, odi file for the test
/// </summary>
   public class ModuleTest{
      public static void Main(){
         string url = "http://127.0.0.1/oz/server.aspx";
         string id = "admin";
         string password = "admin";

         Module m = new Module(url, id, password, true, true);

         IDictionary parameters = new Hashtable();
         parameters["rowcount"] = "10000";

         m.AddODIParameter("대용량 100 만.odi", "대용량.oza", "/대용량",
parameters);

         m.AddApplicationParameter("odi.odinames", "대용량 100 만");
         m.AddApplicationParameter("odi.대용량 100 만.pcount", "1");
         m.AddApplicationParameter("odi.대용량 100 만.args1",
"rowcount=10000");

         Stream ozuFile = m.GetOZU("대용량.oza", "/대용량");

         m = new Module(url, id, password, true, true);

         m.AddODIParameter("parameter_test.odi", "odiparam", "this
is odi parameter");
         m.AddODIParameter("parameter_test.odi", "odiparam2", "this
is  parameter 2");
         m.AddParameter( "formparam", "this is form parameter");

         Stream s = m.GetOZD("parameter_test.ozr", "/",
"http://127.0.0.1/img/TEST.gif");
      }
   }
}
```

Sample : MultiReportSample.cs

```
using System;
using oz.framework.api;

namespace sample{
   public class MultiReportSample{
      public static void Main(){
```

```
          Module maker = new Module("http://127.0.0.1/oz/server.aspx",
"admin", "admin", true, false);

          string[] urls = new string[]{"file://D:\\pictures\\0.gif",
"file://D:\\pictures\\1.gif", "file://D:\\pictures\\2.gif"};

          maker.AddODIParameter("parameter_test.odi", "odiparam",
"testvalue");
          maker.AddParameter("formparam", "testvalue");
          IReportInfo reportInfo = maker.AddReport("parameter_test1.ozr", "/"
, "Report1");
          reportInfo.AddURL(urls);
          reportInfo.AddODIParameter("parameter_test.odi", "odiparam", "form
number 1");
          reportInfo.AddParameter("formparam", "form parameter 1");

          reportInfo = maker.AddReport("parameter_test2.ozr", "/",
"Report2");
          reportInfo.AddURL(urls);
          reportInfo.AddODIParameter("parameter_test.odi", "odiparam", "form
number 2");
          reportInfo.AddParameter("formparam", "form parameter 2");

          reportInfo = maker.AddReport("parameter_test3.ozr", "/",
"Report3");
          reportInfo.AddURL(urls);
          reportInfo.AddODIParameter("parameter_test.odi", "odiparam", "form
number 3");
          reportInfo.AddParameter("formparam", "form parameter 3");

          maker.SaveOZD(@"d:\MultiReport.ozd");
 }
   }
}
```

Class Monitor

Constructor Summary

- **Monitor(string ip, int port, string id, string pw, boolean autoLogin, boolean useUSL)**
- **Monitor (string url, string id, string pw, boolean autoLogin, boolean useUSL)**

Method Summary

- **OZServerInfo GetServerInfo()**
- **MemoryStatus GetMemoryInfo()**
- **Stream DownloadLog()**

Constructor Detail

ASP .NET	`public Monitor(string url, string id, string pw, boolean autoLogin, boolean useUSL)`	
Argument	*url*	오즈 닷넷 서버의 URL ex) string url = "http://127.0.0.1/oz/server.aspx";
	id	사용자 아이디 ex) string id = "admin";
	pw	사용자 패스워드 ex) string pw = "admin";
	autoLogin	자동 로그인 여부 ex) boolean autoLogin = true;
	useUSL	USL 사용 여부 ex) boolean useUSL = false;

Method Detail

■ **GetServerInfo**

Prototype	`public OZServerInfo GetServerInfo() throws OZPAIException`
Definition	서버의 버전 정보와 각 모듈별 버전 정보를 가져옵니다.

■ **GetMemoryInfo**

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

■ **DownloadLog**

rototype	`public stream  DownloadLog() throws OZAPIException`
Definition	서버에 저장된 모니터 로그 파일을 가져옵니다.

관련 Class

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

Server가 실행되고 있는 System의 메모리 상태를 나타냅니다.

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

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 FrameworkVersion : Server가 실행되고 있는 .NET Framework Version
  - public string ServerVersion : 오즈 서버의 버전
  - public string ReleaseNumber : OZ Common Protocol의 릴리즈 버전
  - public int ProtocolNumber : OZ Common Protocol의 프로토콜 버전
  - public string DataModuleReleaseNumber : OZ Data Module의 릴리즈 버전

Sample : MonitorSample.cs

```
using System;

using oz.util;
using oz.framework.api;
using oz.framework.monitor;

namespace sample{
/// <summary>
/// MonitorTest
/// </summary>
   public class MonitorTest{
      public static void Main(){
         string url = "http://127.0.0.1/oz/server.aspx";
         string id = "admin";
         string password = "admin";

         Monitor m = new Monitor(url, id, password, true, true);

         OZServerInfo si = m.GetServerInfo();
         Console.WriteLine(si);

         MemoryStatus ms = m.GetMemoryInfo();
         Console.WriteLine(ms);

         System.IO.Stream monitorLog = m.DownloadLog();
      }
   }
}
```

Class Repository

Constructor Summary

- Repository(string url, string id, string pw, boolean autoLogin, boolean useUSL)

Method Summary

/ / Configuration

- public void SetConfigration(OZAttributeList config)
- public OZAttributeList GetConfiguration()

/ / User

- public int CreateUser(string userName, string pwd, int groupID, string description)
- public void DeleteUser(int userID)
- public void UpdateUserName(int userID, string userName)
- public string GetUserName(int userID)

/ / UserLogin

- public void DisableLogin(string userName)
- public void UpdateLoginDefault(int value)
- public void EnableLogin(string userName)
- public void Logout(int userID)
- public bool IsLoggedIn(int userID)

/ / UserPwd

- public bool CheckPassword(int userID, string password)
- public void UpdatePassword(int userID, string password)

/ / UserDesc

- public void UpdateUserDescription(int userID, string description)

- public string GetUserDescription(int userID)

// UserID

- public int GetGroupID(int userID)
- public int GetUserID(string userName)
- public void UpdateGroupID(int groupID, int userID)

// UserList

- public OZRepositoryUser[] GetUserInfos()
- public OZRepositoryUser[] GetUserInfos(int groupID)
- public OZRepositoryUser[] GetUserInfosOfItem(int itemID, byte Permission)
- public OZRepositoryUser[] GetUserInfosOfCategory(int categoryID, byte Permission)

// Group

- public OZRepositoryUser[] GetUserInfosOfCategory(int categoryID, byte Permission)
- public int CreateGroup(string groupName, int parentGroupID)
- public int CreateGroup(string name, int parentGroupID, string description)
- public void DeleteGroup(int groupID)
- public void UpdateParentGroup(int groupID, int parentGroupID)
- public void UpdateGroupName(int groupID, string groupName)

// GroupAdmin

- public void UpdateGroupAdministrator(int userID, int groupID)
- public bool IsGroupAdministrator(int userID, int groupID)

// GroupList

- public OZRepositoryGroup[] GetChildGroupInfos(int groupID)
- public OZRepositoryGroup[] GetSubGroupInfos(int groupID)
- public OZRepositoryGroup GetGroupInfo(int groupID)
- public OZRepositoryGroup[] GetGroupInfosOfItem(int itemID, byte Permission)
- public OZRepositoryGroup[] GetGroupInfosOfCategory(int categoryID, byte Permission)

// Item

- public int CreateItem(string name, OZItemType type, string description,
- int categoryID, Stream itemData)
- public int CreateItem(string name, OZItemType type, string description,
- string categoryName, Stream itemData)
- public int GetItemID(string name, OZItemType type, int categoryID)
- public int GetItemID(string name, OZItemType type, string categoryName)
- public void DeleteItem(int id)
- public Stream GetItem(int id, int categoryID)
- public void UpdateItemName(int itemID, string itemName)
- public string GetItemPath(int itemID)
- public Stream GetItemDirectly(string name, OZItemType type, string categoryName)
- public Stream GetItemDirectly(string name, OZItemType type, string categoryName, bool compressed)
- public void UpdateItem(int id, Stream itemData)
- public void UpdateItemDirectly(string name, OZItemType type, string categoryName, Stream itemData)
- public bool HasItem(string name, OZItemType type, string categoryName)

// InfoByItem

- public int GetCategoryID(int itemID)
- public void UpdateCategoryID(int categoryID, int newCategoryID, int itemID)

// ItemList

- public OZRepositoryItem[] GetItemInfos()
- public OZRepositoryItem GetItemInfo(int id)
- public OZRepositoryItem[] GetItemInfos(int categoryID)
- public OZRepositoryItem[] GetItemInfos(string categoryName)
- public OZRepositoryItem[] GetItemInfos(int categoryID, int userID, byte Permission)
- public OZRepositoryItem[] GetItemInfos(string categoryName, int userID, byte Permission)
- public OZRepositoryItem[] GetItemInfosOfGroup(int categoryID, int groupID, byte Permission)

- public OZRepositoryItem[] GetItemInfosOfGroup(string categoryName, int groupID, byte Permission)
- public OZRepositoryItem[] GetItemInfosOfUser(int userID, byte Permission)
- public OZRepositoryItem[] GetItemInfosOfGroup(int groupID, byte Permission)

// Category

- public int CreateCategory(string name, int parentCategoryID)
- public int CreateCategory(string categoryPath)
- public void DeleteCategory(int id)
- public int GetCategoryID(string fullPath)
- public void UpdateParentCategory(int id, int parentCategoryID)
- public void UpdateCategoryName(int id, string name)
- public int GetItemCount(int categoryID)
- public OZRepositoryCategory[] GetCategoryInfos(int id)
- public OZRepositoryCategory[] GetCategoryInfosOfUser(int id, int userID, byte Permission)
- public OZRepositoryCategory[] GetCategoryInfosOfGroup(int id, int groupID, byte Permission)
- public OZRepositoryCategory GetCategoryInfo(int id)

// CheckInOut

- public void CheckOut(int itemID, int userID, string checkoutFolder)
- public bool UndoCheckOutItem(int itemID, int userID)
- public bool CheckInItem(bool keepCheckOut, int itemID, int userID, Stream itemData)
- public bool IsCheckOutUser(int itemID, int userID)

// History

- public void RollBackItem(int itemID, int itemVersion)
- public Stream GetItemByVersion(int id, int version)
- public OZRepositoryHistory[] GetHistoryInfos(int itemID)
- public void ClearHistory(int itemID, int version)

Constructor Detail

ASP .NET	public Repository(string url, string id, string pw, boolean autoLogin, boolean useUSL)	
Argument	*url*	오즈 닷넷 서버의 URL ex) string url = "http://127.0.0.1/oz/server.aspx";
	id	사용자 아이디 ex) string id = "admin";
	pw	사용자 패스워드 ex) string pw = "admin";
	autoLogin	자동 로그인 여부 ex) boolean autoLogin = true;
	useUSL	USL 사용 여부 ex) boolean useUSL = false;

Method Detail

/ / Configuration

■ **SetConfiguration**

Prototype	public void SetConfiguration(OZAttributeList config) throws OZAPIException
Definition	Repository 설정을 변경합니다. Repository 타입, 디렉토리별 최대 아이템 개수, 아이템 히스토리의 valid 시간 및 아이템을 압축하여 넣을 것인지 등 %OZ_HOME%conf/repository.properties 에 있는 설정에 대해서 경할 수 있습니다.
Argument	*config* 리파지토리 설정 속성

■ **GetConfiguration**

Prototype	public OZAttributeList GetConfiguration() throws OZAPIException
Definition	Repository의 설정을 가져옵니다. 조회 내용은 Repository 타입, 디렉토리별 최대 아이템 개수, 아이템 히스토리의 valid 일수 등 %OZ_HOME%conf/repository.properties에 설정된 내용입니다.

- Key

  SetConfiguration()과 GetConfiguration()에서 사용되는 key는 다음 표와 같습니

다.

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 password, int groupID, string description) throws OZAPIException`	
Definition	패러미터로 넘기는 이름, 패스워드, 그룹아이디, 사용자설명 값으로 새로운 사용자를 생성하여, 그 사용자의 userid를 return 합니다.	
Argument	*userName*	사용자 이름
	password	패스워드
	groupID	그룹 ID
	description	사용자에 대한 설명 내용

■ DeleteUser

Prototype	`public void DeleteUser(int userID) throws OZAPIException`
Definition	지정한 사용자 ID에 해당하는 사용자의 모든 정보를 삭제합니다.

Argument	*userID*	삭제할 사용자의 ID

■ UpdateUserName

Prototype	public void UpdateUserName(int userID, string userName) throws OZAPIException	
Definition	지정한 사용자 ID에 해당하는 사용자의 이름을 변경합니다.	
Argument	*userID*	사용자 ID
	userName	변경할 사용자 이름

■ GetUserName

Prototype	public string GetUserName(int userID) throws OZAPIException	
Definition	지정한 사용자 ID에 해당하는 사용자의 이름을 가져옵니다.	
Argument	*userID*	사용자 ID

/ / UserLogin

■ DisableLogin

Prototype	public void DisableLogin(string userName) throws OZAPIException	
Definition	지정한 사용자 이름에 해당하는 사용자를 로그인하지 못하도록 설정합니다.	
Argument	*userName*	로그인하지 못하도록 할 사용자 이름

■ UpdateLoginDefault

Prototype	public void UpdateLoginDefault(int value) throws OZAPIException	
Definition	지정한 사용자 ID에 해당하는 사용자로 기본 접속 사용자를 변경합니다.	
Argument	*value*	기본 접속 사용자로 설정할 사용자 ID

■ EnableUserLogin

Prototype	public void EnableUserLogin(string userName) throws OZAPIException	
Definition	지정한 사용자 이름에 해당하는 사용자를 로그인 가능하도록 설정합니다.	
Argument	*userName*	로그인 가능하도록 할 사용자 이름

■ Logout

Prototype	`public void Logout(int userID) throws OZAPIException`
Definition	지정한 사용자 ID에 해당하는 사용자를 로그 아웃 즉, 접속 해제시킵니다.
Argument	*userID* 접속 해제시킬 사용자 ID

■ **IsLoggedIn**

Prototype	`public bool IsLoggedIn(int userID) throws OZAPIException`
Definition	지정한 사용자 ID에 해당하는 사용자가 로그인 상태 즉, 접속 상태인지 확인합니다.
Argument	*userID* 접속 상태를 확인할 사용자 ID

// UserPwd

■ **CheckPassword**

Prototype	`public bool CheckPassword(int userID, string password) throws OZAPIException`	
Definition	사용자의 패스워드가 맞는지 여부를 체크합니다.	
Argument	*userID*	패스워드를 체크할 사용자 ID
	password	체크할 패스워드

■ **UpdatePassword**

Prototype	`public void UpdatePassword(int userID, string password) throws OZAPIException`	
Definition	지정한 사용자 ID에 해당하는 사용자의 패스워드를 변경합니다.	
Argument	*userID*	패스워드를 변경할 사용자 ID
	password	변경할 패스워드

// UserDesc

■ **UpdateUserDescription**

Prototype	`public void UpdateUserDescription(int userID, string description) throws OZAPIException`	
Definition	지정한 사용자 ID에 해당하는 사용자의 사용자 설명 내용을 변경합니다.	
Argument	*userID*	사용자 설명 내용을 변경할 사용자 ID
	description	변경할 사용자 설명

■ **GetUserDescription**

Prototype	public string GetUserDescription(int userID) throws OZAPIException
Definition	지정한 사용자 ID에 해당하는 사용자의 사용자 설명 내용을 가져옵니다.
Argument	*userID* 사용자 설명 내용을 가져올 사용자 ID

// UserID

■ **GetGroupID**

Prototype	public int GetGroupID(int userID) throws OZAPIException
Definition	지정한 사용자 ID에 해당하는 사용자의 그룹 ID를 가져옵니다.
Argument	*userID* 그룹 ID를 가져올 사용자 ID

■ **GetUserID**

Prototype	public int GetUserID(string userName) throws OZAPIException
Definition	지정한 사용자 이름에 해당하는 사용자의 ID를 가져옵니다.
Argument	*userName* 사용자 ID를 가져올 사용자 이름

■ **UpdateGroupID**

Prototype	public void UpdateGroupID(int groupID, int userID) throws OZAPIException	
Definition	사용자의 그룹 ID를 변경합니다.	
Argument	*groupID*	변경할 그룹 ID
	userID	그룹 ID를 변경할 사용자 ID

// UserList

■ **GetUserInfos**

Prototype	public OZRepositoryUser[] GetUserInfos() throws OZAPIException
Definition	서버에 등록된 사용자의 전체 리스트를 OZRepositoryUser 객체로 가져옵니다.

■ **GetUserInfos**

Prototype	public OZRepositoryUser[] GetUserInfos(int groupID) throws OZAPIException
Definition	지정한 그룹 ID에 등록되어 있는 모든 사용자의 정보를 가져옵니다.

Argument	*groupID*	사용자의 정보를 가져올 그룹 ID

■ **GetUserInfosOfItem**

Prototype	`public OZRepositoryUser[] GetUserInfosOfItem(int itemID, byte permission) throws OZAPIException`
Definition	지정한 아이템 ID에 대해 Permission 이상의 권한을 가지고 있는 모든 사용자 정보를 가져옵니다.
Argument	*itemID* 아이템 ID
	permission 권한 정보

■ **GetUserInfosOfCategory**

Prototype	`public OZRepositoryUser[] GetUserInfosOfCategory(int categoryID, byte permission) throws OZAPIException`
Definition	지정한 카테고리 ID에 대해 Permission 이상의 권한을 가지고 있는 모든 사용자 정보를 가져옵니다.
Argument	*categoryID* 카테고리 ID
	permission 권한 정보

/ / Group

■ **CreateGroup**

Prototype	`public int CreateGroup(string groupName, int parentGroupID) throws OZAPIException`
Definition	새로운 그룹을 생성하고, 생성된 그룹의 ID를 반환합니다.
Argument	*groupName* 새로 생성할 그룹의 이름
	parentGroupID 새로 생성할 그룹의 상위 그룹 ID

■ **DeleteGroup**

Prototype	`public void DeleteGroup(int groupID) throws OZAPIException`
Definition	지정한 그룹 ID에 해당하는 그룹을 삭제합니다.
Argument	*groupID* 삭제할 그룹 ID

■ **UpdateParentGroup**

Prototype	`public void UpdateParentGroup(int groupID, int parentGroupID) throws OZAPIException`
Definition	지정한 그룹의 상위 그룹 ID에 해당하는 그룹의 이름을 변경합니다.

Argument	*groupID*	상위 그룹의 이름을 변경할 그룹 ID
	parentGroupID	변경할 그룹의 이름

■ **UpdateGroupName**

Prototype	`public void UpdateGroupName(int groupID, string groupName) throws OZAPIException`	
Definition	지정한 그룹 ID에 해당하는 그룹의 이름을 변경합니다.	
Argument	*groupID*	그룹의 이름을 변경할 그룹 ID
	groupName	변경할 그룹의 이름

/ / GroupAdmin

■ **UpdateUserGroupAdmin**

Prototype	`public void UpdateGroupAdministrator(int userID, int groupID) throws OZAPIException`	
Definition	지정한 그룹 ID에 해당하는 그룹의 관리자를 지정한 사용자로 변경합니다.	
Argument	*userID*	관리자를 변경할 그룹 ID
	groupID	새로운 관리자로 설정할 사용자 ID

■ **IsGroupAdministrator**

Prototype	`public bool IsGroupAdministrator(int userID, int groupID) throws OZAPIException`	
Definition	지정한 사용자 ID에 해당하는 사용자가 해당 그룹의 관리자인지 여부를 확인합니다.	
Argument	*userID*	관리자인지 체크할 사용자 ID
	groupID	그룹 ID

/ / GroupList

■ **GetChildGroupInfos**

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

■ **GetSubGroupInfos**

Prototype	`public OZRepositoryGroup[] GetSubGroupInfos(int groupID) throws OZAPIException`
Definition	지정한 그룹 ID에 속한 모든 하위 그룹의 정보를 가져옵니다. (not recursively)
Argument	*groupID* 하위 그룹의 정보를 가져올 그룹 ID

■ **GetGroupInfo**

Prototype	`public OZRepositoryGroup GetGroupInfo(int groupID) throws OZAPIException`
Definition	지정한 그룹 ID에 해당하는 그룹의 정보를 가져옵니다.
Argument	*groupID* 그룹 ID

■ **GetGroupInfosOfItem**

Prototype	`public OZRepositoryGroup[] GetGroupInfosOfItem(int itemID, byte permission) throws OZAPIException`	
Definition	지정한 아이템 ID에 대해 permission 이상의 권한을 가지고 있는 모든 그룹 정보를 가져옵니다.	
Argument	*itemID*	아이템 ID
	permission	권한 정보

■ **GetGroupInfosOfCategory**

Prototype	`public OZRepositoryGroup[] GetGroupInfosOfCategory(int categoryID, byte permission) throws OZAPIException`	
Definition	지정한 카테고리 ID에 대해 Permission 이상의 권한을 가지고 있는 모든 그룹 정보를 가져옵니다.	
Argument	*categoryID*	카테고리 ID
	permission	권한 정보

// Item

■ **CreateItem**

Prototype	`public int CreateItem(string itemName, OZItemType itemType, string itemDescription, int categoryID, Stream itemData) throwcs OZAPIException` `public int CreateItem(string itemName, OZItemType itemType, string itemDescription, string categoryName, Stream itemData) throwcs OZAPIException`

Definition	아이템 이름, 타입, desc. 아이템이 속할 카테고리 ID, 아이템 파일을 패러미터로 넘기고, 새로운 아이템을 생성하여, 그 item ID를 반환합니다.	
Argument	*itemName*	새로 생성할 아이템 이름
	itemType	새로 생성할 아이템 타입
	itemDescription	새로 생성할 아이템 설명
	categoryID	새로 생성할 아이템의 카테고리 ID
	itemData	새로 생성할 아이템의 입력 스트림
	categoryName	새로 생성할 아이템의 카테고리 이름

■ GetItemId

Prototype	`public int GetItemID(string itemName, OZItemType itemType, int categoryID) throwcs OZAPIException` `public int GetItemID(string itemName, OZItemType itemType, string categoryName) throwcs OZAPIException`	
Definition	패러미터로 넘기는 아이템명, 아이템 타입, 아이템이 속한 카테고리 ID에 해당하는 그 아이템의 ID를 가져옵니다. ItemType은 관련 클래스인 OZItemInfo를 참조하시기 바랍니다.	
Argument	*itemName*	가져올 아이템 이름
	itemType	가져올 아이템 타입
	categoryID	가져올 아이템의 카테고리 ID
	categoryName	가져올 아이템의 카테고리 이름

■ DeleteItem

Prototype	`public void deleteItem(int itemID) throws OZAPIException`	
Definition	지정한 아이템을 리파지토리에서 삭제합니다.	
Argument	*itemID*	삭제할 아이템 ID

■ GetItem

Prototype	`public Stream getItem(int itemID) throws OZAPIException`	
Definition	지정한 아이템 ID에 해당하는 아이템을 가져옵니다.	
Argument	*itemID*	가져올 아이템 ID

■ UpdateItemName

Prototype	`public void UpdateItemName(int itemID, string itemName) throws OZAPIException`

Definition	지정한 아이템 ID에 해당하는 아이템의 이름을 변경합니다.	
Argument	*itemID*	이름을 변경할 아이템 ID
	itemName	변경할 아이템 이름

■ **GetItemDirectly**

Prototype	public Stream GetItemDirectly(string itemName, OZItemType itemType, string categoryName) throws OZAPIException public Stream GetItemDirectly(string itemName, OZItemType itemType, string categoryName, bool compressedItem) throws OZAPIException	
Definition	아이템을 가져옵니다.	
Argument	*itemName*	가져올 아이템 이름
	itemType	가져올 아이템 타입
	categoryName	가져올 아이템의 카테고리 이름
	compressedItem	가져올 아이템의 압축 여부

■ **UpdateItem**

Prototype	public void updateItem(int itemID, stream itemData) throws OZAPIException	
Definition	지정한 아이템 ID에 해당하는 아이템을 변경합니다.	
Argument	*itemID*	변경할 아이템 ID
	itemData	변경할 내용

■ **UpdateItemDirectly**

Prototype	public void UpdateItemDirectly(string itemName, OZItemType itemType, string categoryName, Stream itemData) throws OZAPIException	
Definition	지정한 아이템 ID에 해당하는 아이템을 변경합니다.	
Argument	*itemName*	변경할 아이템 이름
	itemType	변경할 아이템 타입
	categoryName	변경할 아이템의 카테고리 이름
	itemData	변경할 내용

■ **HasItem**

Prototype	public bool HasItem(string itemName, OZItemType itemType, string categoryName) throws OZAPIException	

Definition	지정한 아이템이 해당 카테고리에 있는지 여부를 확인합니다.	
Argument	*itemName*	체크할 아이템 이름
	itemType	체크할 아이템 타입
	categoryName	체크할 카테고리 이름

// InfoByItem

■ **GetCategoryID**

Prototype	`public int GetCategoryID(int itemID) throws OZAPIException`
Definition	지정한 아이템 ID에 해당하는 아이템이 속한 카테고리 ID를 가져옵니다.
Argument	*itemID* 아이템 ID

■ **UpdateCategoryID**

Prototype	`public void UpdateCategoryID(int categoryID, int newCategoryID, int itemID) throws OZAPIException`	
Definition	지정한 아이템 ID에 해당하는 아이템의 카테고리를 변경합니다.	
Argument	*categoryID*	변경할 카테고리 ID
	newCategoryID	변경될 새 카테고리 ID
	itemID	카테고리를 변경할 아이템 ID

// ItemList

■ **GetItemInfos**

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

■ **GetItemInfo**

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

■ **GetItemInfos**

Prototype	public OZRepositoryItem[] GetItemInfos(int categoryID) throws OZAPIException public OZRepositoryItem[] GetItemInfos(string categoryFullPath) throws OZAPIException public OZRepositoryItem[] GetItemInfos(int categoryID, int userID, byte permission) throws OZAPIException public OZRepositoryItem[] GetItemInfos(string categoryFullPath, int userID, byte permission) throws OZAPIException
Definition	지정한 옵션에 해당하는 모든 아이템의 정보를 가져옵니다.
Argument	*categoryID* 가져올 아이템의 카테고리 ID
	categoryFullPath 가져올 아이템의 카테고리 전체경로
	userID 가져올 아이템의 사용자 ID
	permission 권한 정보

■ **GetItemInfosOfGroup**

Prototype	public OZRepositoryItem[] GetItemInfosOfGroup(int categoryID, int groupID, byte Permission) throws OZAPIException public OZRepositoryItem[] GetItemInfosOfGroup(string categoryFullPath, int groupID, byte permission) throws OZAPIException
Definition	지정한 옵션에 해당하는 아이템의 정보를 가져옵니다.
Argument	*categoryID* 가져올 아이템의 카테고리 ID
	groupID 가져올 아이템의 그룹 ID
	permission 권한 정보
	categoryFullPath 가져올 아이템의 카테고리 전체 경로

■ **GetItemInfosOfUser**

Prototype	public OZRepositoryItem[] GetItemInfosOfUser(int userID, byte permission) throws OZAPIException
Definition	지정한 사용자 ID에 해당하는 사용자에게 Permission 이상의 권한이 부여된 모든 아이템의 정보를 가져옵니다.
Argument	*userID* 사용자 ID
	permission 권한 정보

■ **GetItemInfosOfGroup**

Prototype	public OZRepositoryItem[] GetItemInfosOfGroup(int groupID, byte permission) throws OZAPIException	
Definition	지정한 그룹 ID에 해당하는 그룹에게 Permission 이상의 권한이 부여된 모든 아이템의 정보를 가져옵니다.	
Argument	*groupID*	그룹 ID
	permission	권한 정보

/ / Category

■ **CreateCategory**

Prototype	public int CreateCategory(string categoryName, int parentCategoryID) throws OZAPIException	
Definition	카테고리를 새로 생성하고, 생성된 카테고리의 ID를 반환합니다.	
Argument	*categoryName*	생성할 카테고리의 이름
	parentCategoryID	생성할 카테고리의 상위 카테고리 ID

■ **CreateCategory**

Prototype	public int CreateCategory(string categoryPath) throws OZAPIException	
Definition	지정된 경로로 카테고리를 새로 생성하고, 생성된 카테고리의 ID를 반환합니다.	
Argument	*categoryPath*	새로 생성할 카테고리 전체 경로

■ **DeleteCategory**

Prototype	public void DeleteCategory(int categoryID) throws OZAPIException	
Definition	지정한 카테고리 ID에 해당하는 카테고리를 삭제합니다.	
Argument	*categoryID*	삭제할 카테고리 ID

■ **GetCategoryID**

Prototype	public int GetCategoryID(string fullPath) throws OZAPIException	
Definition	전체 경로에 해당하는 카테고리의 ID를 가져옵니다.	
Argument	*fullPath*	카테고리의 ID를 가져올 카테고리 경로

UpdateParentCategory

Prototype	public void UpdateParentCategory(int categoryID, int parentCategoryID) throws OZAPIException	
Definition	지정한 카테고리의 상위 카테고리 ID를 변경합니다.	
Argument	*categoryID*	카테고리 ID
	parentCategoryID	상위 카테고리 ID

UpdateCategoryName

Prototype	public void UpdateCategoryName(int categoryID, string categoryName) throws OZAPIException	
Definition	지정한 카테고리 ID에 해당하는 카테고리의 이름을 변경합니다.	
Argument	*categoryID*	이름을 변경할 카테고리 ID
	categoryName	변경할 카테고리 이름

GetItemCount

Prototype	public int GetItemCount(int categoryID) throws OZAPIException	
Definition	지정한 카테고리에 속해 있는 모든 아이템의 개수를 가져옵니다.	
Argument	*categoryID*	아이템의 개수를 얻을 카테고리 ID

GetCategoryInfos

Prototype	public OZRepositoryCategory[] GetCategoryInfos(int categoryID) throws OZAPIException	
Definition	지정한 카테고리에 포함된 모든 하위 카테고리의 정보를 가져옵니다.	
Argument	*categoryID*	카테고리 ID

GetCategoryInfo

Prototype	public OZRepositoryCategory GetCategoryInfo(int categoryID) throws OZAPIException	
Definition	지정한 카테고리의 정보를 가져옵니다.	
Argument	*categoryID*	정보를 가져올 카테고리 ID

GetCategoryInfosOfUser

Prototype	public OZRepositoryCategory[] GetCategoryInfosOfUser(int categoryID, int userID, byte Permission) throws OZAPIException

Definition	지정한 사용자 ID에 해당하는 사용자에게 permission 이상의 권한이 부여된 모든 카테고리의 정보를 가져옵니다.	
Argument	*userID*	사용자 ID
	categoryID	카테고리 ID
	Permission	권한 정보

■ **GetCategoryInfosOfGroup**

Prototype	`public OZRepositoryCategory[] GetCategoryInfosOfGroup(int groupid, int categorypID, byte permission) throws OZAPIException`
Definition	지정한 그룹 ID에 해당하는 그룹에게 permission 이상의 권한이 부여된 모든 카테고리의 정보를 가져옵니다.
Argument	*groupID* 그룹 ID
	categoryID 카테고리 ID
	permission 권한 정보

/ / CheckInOut

■ **checkOut**

Prototype	`public void checkOut(int itemID, int userID, string checkOutFolder) throws OZAPIException`
Definition	지정한 아이템을 지정한 사용자 ID로 체크 아웃할 폴더에 체크 아웃을 합니다.
Argument	*itemID* 체크 아웃할 아이템 ID
	userID 체크 아웃할 사용자 ID
	checkOutFolder 체크 아웃할 폴더 이름

■ **undoCheckOut**

Prototype	`public void undoCheckOut(int itemID, int userID) throws OZAPIException`
Definition	지정한 아이템을 지정한 사용자 ID로 체크 아웃 취소합니다.
Argument	*itemID* 체크 아웃 취소할 아이템 ID
	userID 체크 아웃 취소할 사용자 ID

■ **checkIn**

Prototype	`public void checkIn(boolean keepChkOut, int itemId, int userID, stream itemData) throws OZAPIException`

Definition	지정한 아이템을 지정한 사용자 ID로 체크인합니다.	
Argument	*keepChkOut*	체크 아웃 상태를 유지할지 여부
	itemID	체크인할 아이템 ID
	userID	체크인할 사용자 ID
	itemData	아이템의 입력 스트림

■ **isCheckOutUser**

Prototype	`public boolean isCheckOutUser(int itemID, int userID) throws OZAPIException`
Definition	지정한 사용자가 지정한 아이템을 체크 아웃했는지 여부를 확인합니다.
Argument	*itemID* 아이템 ID
	userID 체크 아웃한 사용자인지 체크할 사용자 ID

/ / History

■ **GetItemByVersion**

Prototype	`public Stream GetItemByVersion(int itemID, int itemVersion) throws OZAPIException`
Definition	지정한 아이템 ID에 해당하는 아이템 중 지정한 버전의 아이템을 가져옵니다.
Argument	*itemID* 가져올 아이템 ID
	itemVersion 가져올 아이템 버전

■ **GetHistoryInfos**

Prototype	`public OZRepositoryHistory[] GetHistoryInfos(int itemID) throws OZAPIException`
Definition	지정한 아이템의 히스토리 정보를 가져옵니다.
Argument	*itemID* 히스토리 정보를 가져올 아이템의 ID

■ **ClearHistory**

Prototype	`public void clearHistory(int itemID, int itemVersion) throws OZAPIException`
Definition	지정한 아이템에 대해 특정 버전의 히스토리를 삭제합니다.
Argument	*itemID* 히스토리를 삭제할 아이템 ID
	itemVersion 히스토리를 삭제할 아이템 버전

■ **RollBack**

Prototype	public void rollBack(int itemID, int itemVersion) throws OZAPIException	
Definition	지정한 아이템을 지정한 버전으로 복원시킵니다.	
Argument	*itemID*	복원시킬 아이템 ID
	itemVersion	복원시킬 아이템 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.

- 속성

▪ Name

Prototype	public string Name{get;}
Definition	사용자 이름

▪ ID

Prototype	public int ID{get;}
Definition	사용자 ID

▪ GroupList

Prototype	public System.Collections.IList GroupList {get;}
Definition	그룹 정보

▪ Description

Prototype	public string Description{get;}
Definition	사용자 설명 내용

▪ PassWord

Prototype	public string PassWord{get;}
Definition	사용자 패스워드

▪ Permission

Prototype	public byte Permission{get;}

Definition	사용자 권한 정보 • 0 : None(권한 없음) • 1 : View(목록 조회만 가능) • 3 : Read(아이템 읽기 가능) • 7 : Write(아이템 삭제 가능)

▪ DirectPermission

Prototype	`public byte DirectPermission{get;}`
Definition	사용자 권한 정보

▪ InDirectPermission

Prototype	`public byte InDirectPermission{get;}`
Definition	사용자 권한 정보

▪ IsLoggedIn

Prototype	`public bool IsLoggedIn{get;}`
Definition	사용자가 서버에 접속되어 있는지 여부

▪ SessionID

Prototype	`public int SessionID{get;}`
Definition	사용자의 세션 ID

▪ IsLoginEnabled

Prototype	`public bool IsLoginEnabled{get;}`
Definition	사용자가 접속 가능 여부

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

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

- 속성

▪ Name

Prototype	`public string Name{get;}`
Definition	그룹 이름

▪ ID

Prototype	`public int ID{get;}`
Definition	그룹 ID

- ParentID

Prototype	`public int ParentID{get;}`
Definition	상위 그룹 ID

- GroupAdmininistratorList

Prototype	`public System.Collections.IList GroupAdministratorList {get;}`
Definition	그룹 관리자 정보

- DirectPermission

Prototype	`public byte DirectPermission{get;}`
Definition	그룹 권한 정보

- InDirectPermission

Prototype	`public byte InDirectPermission{get;}`
Definition	그룹 권한 정보

- Permission

Prototype	`public byte Permission{get;}`
Definition	그룹 권한 정보

- Description

Prototype	`public string Description{get;}`
Definition	그룹 설명 내용

- FullPath

Prototype	`public string FullPath{get;}`
Definition	그룹 전체 경로

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

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

- 속성
  - Name

Prototype	`public string Name{get;}`
Definition	아이템 이름

  - ID

Prototype	`public int ID{get;}`
Definition	아이템 ID

  - Type

Prototype	`public OZItemType Type{get;}`
Definition	아이템 타입 `enum OZItemType{ ODI, OZR, SDM, USDM, OZD, IMG }`

  - Description

Prototype	`public string Description{get;}`
Definition	아이템 설명 내용

  - CheckOutUserID

Prototype	`public int CheckOutUserID{get;}`
Definition	체크 아웃한 사용자 ID

  - CheckOutUserName

Prototype	`public string CheckOutUserName{get;}`
Definition	체크 아웃한 사용자 이름

  - CheckOutFolder

Prototype	`public string CheckOutFolder{get;}`
Definition	체크 아웃한 아이템의 폴더 이름

  - UpdateTime

Prototype	`public string UpdateTime{get;}`
Definition	아이템을 갱신한 시간 정보

  - IsCheckedOut

Prototype	`public bool IsCheckedOut{get;}`
Definition	체크 아웃 여부

- DirectPermission

Prototype	`public byte DirectPermission{get;}`
Definition	권한 정보

- InDirectPermission

Prototype	`public byte InDirectPermission{get;}`
Definition	권한 정보

- AdministratorList

Prototype	`public System.Collections.IList AdministratorList{get;}`
Definition	관리자 정보

- CategoryList

Prototype	`System.Collections.IList CategoryList{get;}`
Definition	카테고리 정보

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

오즈 리파지토리의 카테고리 대한 기본적인 정보를 가지고 있는 클래스입니다.

- 속성

- Name

Prototype	`public string Name{get;}`
Definition	카테고리 이름

- ID

Prototype	`public int ID{get;}`
Definition	카테고리 ID

- ParentID

Prototype	`public int ParentID{get;}`
Definition	상위 카테고리의 ID

- CategoryAdministratorList

Prototype	public System.Collections.IList CategoryAdministratorList {get;}
Definition	카테고리의 관리자 정보

- DirectPermission

Prototype	public byte DirectPermission{get;}
Definition	권한 정보

- InDirectPermission

Prototype	public byte InDirectPermission{get;}
Definition	권한 정보

- Permission

Prototype	public byte Permission{get;}
Definition	권한 정보

- Description

Prototype	public string Description{get;}
Definition	카테고리의 설명 내용

- FullPath

Prototype	public string FullPath{get;}
Definition	카테고리의 전체 경로

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

오즈 아이템 히스토리의 정보를 가지고 있는 클래스입니다.

- 속성

- ItemPath

Prototype	public string ItemPath{get;}
Definition	아이템의 경로

- ItemVersion

Prototype	public int ItemVersion{get;}
Definition	아이템의 버전 정보

- Date

Prototype	`public string Date{get;}`
Definition	아이템의 히스토리 날짜 정보

- CheckInUser

Prototype	`public string CheckInUser{get;}`
Definition	아이템의 체크인 사용자 정보

Sample : RepositorySample.cs

```
using System;
using System.IO;
using System.Reflection;
using System.Collections;

using oz.util;
using oz.framework.api;
using oz.framework.repository;

namespace sample{
/// <summary>
/// RepositoryTest
///
/// Before start
/// You should modify file open logic.
/// In this sample we handle project resource whose id is
"sample.parameter_test.odi"
/// as a file to upload
/// </summary>
   public class RepositoryTest{
      private static Repository s_repository = null;

      public static void Main(){
         string url = "http://127.0.0.1/oz/server.aspx";
         string id = "admin";
         string password = "admin";

         s_repository = new Repository(url, id, password, true, true);

         repositoryConfiguration();

         categoryTest();
         itemTest();
         groupTest();
```

```
         userTest();
      }

   private static void repositoryConfiguration(){
      OZAttributeList attrs = s_repository.GetConfiguration();
      Console.WriteLine(attrs);

      s_repository.SetConfiguration(attrs);
   }

   private static void historyTest(int itemID){
   const int version = 0;

   // you can store item using this input stream
   Stream input = s_repository.GetItemByVersion(itemID, version);

   OZRepositoryHistory[] historyInfos =
      s_repository.GetHistoryInfos(itemID);
   foreach(OZRepositoryHistory history in historyInfos){
      Console.WriteLine(history);
   }
   s_repository.RollBack(itemID, version);
   }

   private static void checkInOutTest(int itemID){
      int userID = s_repository.GetUserID("admin");
      string checkoutFolder = ".";
      s_repository.CheckOut(itemID, userID, checkoutFolder);

      s_repository.UndoCheckOut(itemID, userID);
      s_repository.CheckOut(itemID, userID, checkoutFolder);

      Assembly asm = Assembly.GetExecutingAssembly();
      Stream item = asm.GetManifestResourceStream
("sample.parameter_test.odi");
      try{
         s_repository.CheckIn(false, itemID, userID, item);
      }
      finally{
         item.Close();
      }
   }


   private static void categoryTest(){
      int userID = s_repository.GetUserID("admin");
      int groupID = s_repository.GetGroupID(userID);
      int categoryID = s_repository.CreateCategory("/Poultry");
      Console.WriteLine("Created category : {0}", categoryID);
```

```
   int childCategoryID = s_repository.CreateCategory("Chickens", cat
egoryID);
      Console.WriteLine("Created child category : {0}", childCategoryID);

      s_repository.DeleteCategory(childCategoryID);

      s_repository.UpdateCategoryName(categoryID, "Fishes");

      int anotherCategoryID = s_repository.CreateCategory("/Category Test");
      s_repository.UpdateParentCategory(categoryID, anotherCategoryID);

      Console.WriteLine("Item count in category [{0}] : {1}", categoryID,
      s_repository.GetItemCount(categoryID));

      OZRepositoryCategory[] categoryInfos =
      s_repository.GetCategoryInfos(anotherCategoryID);
      foreach(OZRepositoryCategory category in categoryInfos){
         Console.WriteLine(category);
      }

      categoryInfos = s_repository.GetCategoryInfosOfUser(anotherCategoryID,
userID, 2);
      foreach(OZRepositoryCategory category in categoryInfos){
         Console.WriteLine(category);
      }

      s_repository.DeleteCategory(categoryID);
      s_repository.DeleteCategory(anotherCategoryID);
   }

   private static void itemListTest(int itemID){
      int userID = s_repository.GetUserID("admin");
      int groupID = s_repository.GetGroupID(userID);
      string categoryName = "/";
      int categoryID = s_repository.GetCategoryID(categoryName);

      OZRepositoryItem itemInfo = s_repository.GetItemInfo(itemID);
      Console.WriteLine(itemInfo);

      OZRepositoryItem[] itemInfos = s_repository.GetItemInfos(categoryID);
      foreach(OZRepositoryItem ii in itemInfos)
      Console.WriteLine(ii);

      itemInfos = s_repository.GetItemInfos(categoryName, userID, 2);
      foreach(OZRepositoryItem ii in itemInfos)
         Console.WriteLine(ii);

      itemInfos = s_repository.GetItemInfosOfGroup(categoryName, groupID,
2);
      foreach(OZRepositoryItem ii in itemInfos)
```

```
            Console.WriteLine(ii);
    }

    private static void itemTest(){
        string itemName = "api test.odi";
        string categoryName = "/api test";
        string description = "item upload test";

        int categoryID = s_repository.CreateCategory(categoryName);

        Stream item =
Assembly.GetExecutingAssembly().GetManifestResourceStream
("sample.parameter_test.odi");
        int itemID;
        try{
            itemID = s_repository.CreateItem(itemName, OZItemType.ODI,
                description, categoryID, item);
            Console.WriteLine("Item uploaded : {0}", itemID);
        }
        finally{
            item.Close();
        }

        itemName = "Changed item name.odi";
        s_repository.UpdateItemName(itemID, itemName);
        Console.WriteLine("Changed item name : {0}",
s_repository.GetItemInfo(itemID).Name);

        itemListTest(itemID);
        checkInOutTest(itemID);
        categoryTest();
        historyTest(itemID);

        s_repository.DeleteItem(itemID);
        s_repository.DeleteCategory(categoryID);
    }

    private static void groupListTest(int gropuID){
        int userID = s_repository.GetUserID("admin");
        int groupID = s_repository.GetGroupID(userID);
        OZRepositoryGroup gi= s_repository.GetGroupInfo(groupID);
        Console.WriteLine(gi);
    }

    private static void groupAdministratorTest(int groupID){
        IList admins =
s_repository.GetGroupInfo(groupID).GroupAdministratorList;
    Console.WriteLine(admins[0].ToString());

    int userID = s_repository.CreateUser("test id", "1234567", groupID, "");
```

```
   s_repository.UpdateGroupAdministrator(userID, groupID);
   admins = s_repository.GetGroupInfo(groupID).GroupAdministratorList;
   Console.WriteLine(admins[0].ToString());

   s_repository.DeleteUser(userID);
   }

   private static void groupTest(){
      int adminID = s_repository.GetUserID("admin");
      int rootGroupID = s_repository.GetGroupID(adminID);
      string groupName = "forcs";
      int groupID = s_repository.CreateGroup(groupName, rootGroupID, "test
group");

      int tempGroupID = s_repository.CreateGroup("Temporary group",
rootGroupID);

      s_repository.UpdateParentGroup(groupID, tempGroupID);

      s_repository.UpdateParentGroup(groupID, rootGroupID);
      s_repository.DeleteGroup(tempGroupID);

      s_repository.UpdateGroupName(groupID, "OZ XStudio");

      groupAdministratorTest(groupID);
      groupListTest(groupID);

      s_repository.DeleteGroup(groupID);
   }


      private static void userTest(){
         string userName = "forcs";
         string password = "111111";
         string description = "test account";

         int groupID =
s_repository.GetGroupID(s_repository.GetUserID("admin"));
         int userID = s_repository.CreateUser(userName, password, groupID,
description);

         int prevGroupID = s_repository.GetGroupID(userID);
         int newGroupID = s_repository.CreateGroup("Group for test",
prevGroupID);

         s_repository.UpdateGroupID(newGroupID, userID);

         userName = s_repository.GetUserName(userID);
```

```
            s_repository.UpdateGroupID(prevGroupID, userID);
            s_repository.DeleteGroup(newGroupID);

            description = s_repository.GetUserDescription(userID);

            Console.WriteLine("Password matches? " +
s_repository.CheckPassword(userID, "new password"));

            s_repository.UpdatePassword(userID, "new password");

            Console.WriteLine("Password matches? " +
s_repository.CheckPassword(userID, "new password"));

            s_repository.UpdateLoginDefault(userID);

            userName = s_repository.GetUserName(userID);
            s_repository.DisableLogin(userName);
            s_repository.EnableLogin(userName);

            int categoryID = s_repository.CreateCategory("/User list test");

            OZRepositoryUser[] userInfos = s_repository.GetUserInfos();
            foreach(OZRepositoryUser ui in userInfos)
            Console.WriteLine(ui);

            userInfos = s_repository.GetUserInfos(groupID);
            foreach(OZRepositoryUser ui in userInfos)
             Console.WriteLine(ui);

            s_repository.DeleteCategory(categoryID);

            s_repository.DeleteUser(userID);
        }
    }
}
```

Ⅱ. RDB Store for .NET

- DataAction Store DataAction 인터페이스
- DataAction Store DataAction 구현

RDB Store DataAction 인터페이스

오즈에서 제공하는 RDB 스토어에 대한 DataAction 인터페이스와 애트리뷰트는 다음과 같습니다.

■ 인터페이스 방식

```
public interface IRDBDelegate : IDelegate
{
   System.Data.IDbConnection Connection{ set; }

   void Init(string @param);
   void Close();

   string            Insert(oz.framework.dac.OZDACItem           dac,
System.Collections.IDictionary parameters);
   string            Update(oz.framework.dac.OZDACItem           dac,
System.Collections.IDictionary parameters);
   string            Delete(oz.framework.dac.OZDACItem           dac,
System.Collections.IDictionary parameters);

   string Commit();
   void Rollback();
}
```

※ 참고사항 : 사용자 DataAction 기능 구현 시 함수를 호출하는 순서는 다음과 같습니다.

1. Connection()
2. Init()
3. Insert(), Update(), Delete()
4. Commit(), Rollback()
5. Close()

■ 애트리뷰트 방식

- RDBDelegateAttribute

```
[AttributeUsage(AttributeTargets.Class, AllowMultiple=false)]
public sealed class RDBDelegateAttribute : Attribute
{
}
```

RDB Store DataAction 구현

DataAction 인터페이스 예제

다음은 사용자 정의 DataAction 인터페이스를 구현한 예제입니다.

```
using System;
using System.Data;
using System.Collections;

namespace com.forcs.sample.dataaction.rdb
{
   /// <summary>
      /// 호출 순서는 아래와 같습니다.
      /// ILoggerRef의 메서드
      /// IRDBDelegate.Connection{ set; }
      /// IRDBDelegate.Init()
      /// IRDBDelegate.Insert() or Delete() or Update()
      /// IRDBDelegate.Commit() or Rollback()
      /// IRDBDelegate.Close()
   /// </summary>
      public class DelegateSample : oz.udd.IRDBDelegate, oz.udd.ILoggerRef
   {
      private oz.framework.log.OZLog log;
      private IDbConnection _con;
      public DelegateSample()
                                                                                {
  }

      #region IRDBDelegate 멤버
      public System.Data.IDbConnection Connection
      {
         set
         {
            //DB연결 스크립트를 추가합니다.
            _con = value;
         }
      }

      #endregion
      #region IDelegate 멤버
      public void Rollback()
      {
         //Rollback 스크립트를 추가합니다.
      }
```

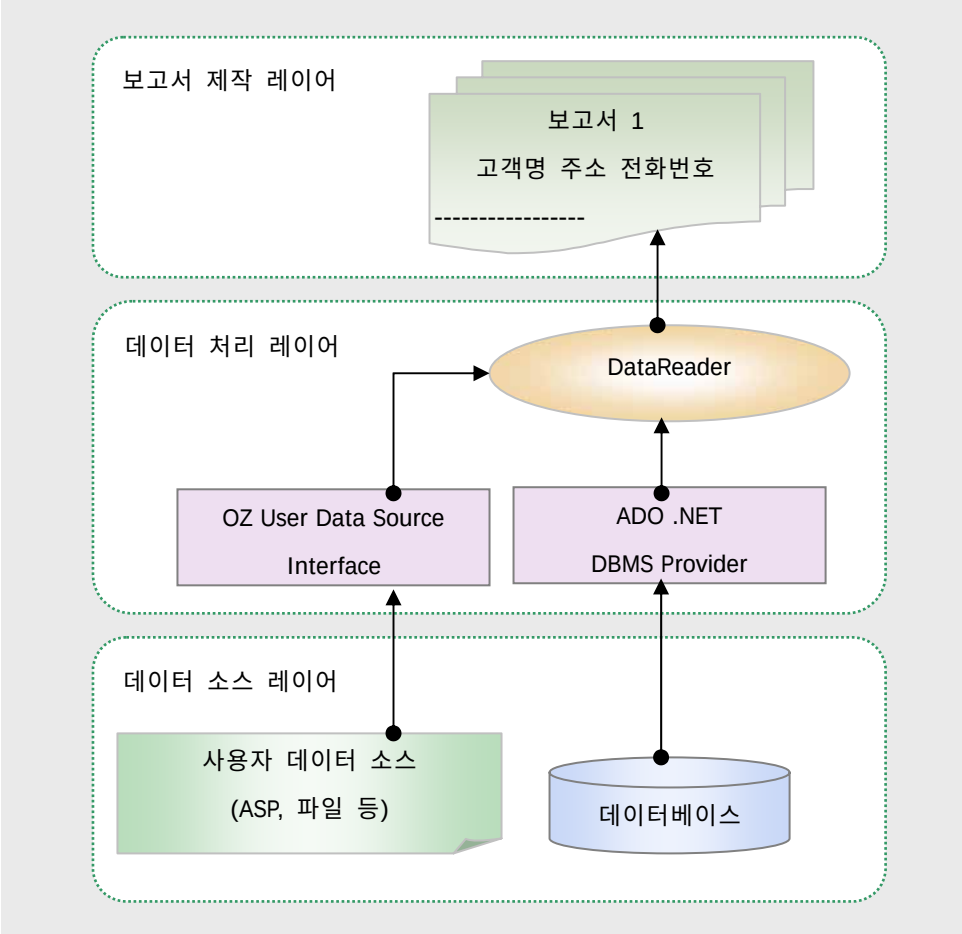
```
        public string Commit()
        {
            //Commit하는 스크립트를 추가합니다.
        }

        public void Init(string param)
        {
            // TODO:  DelegateSample.Init 구현을 추가합니다.
            log.Debug("Initialization called. Parameter is '" + param + "'");
        }

        public void Close()
        {
            //Close 하는 스크립트를 추가합니다.
        }

        public          string          Insert(oz.framework.dac.OZDACItem          dac,
System.Collections.IDictionary parameters)
        {
            //Insert하는 스크립트를 추가합니다.
        }

        public          string          Delete(oz.framework.dac.OZDACItem          dac,
System.Collections.IDictionary parameters)
        {
            //Delete하는 스크립트를 추가합니다.
        }

        public          string          Update(oz.framework.dac.OZDACItem          dac,
System.Collections.IDictionary parameters)
        {
            //Update하는 스크립트를 추가합니다.
        }

        #endregion
        #region ILoggerRef 멤버
        public oz.framework.log.OZLog Log
        {
            set
            {
                //log를 기록하는 스크립트를 추가합니다.
                log = value;
            }
        }
        #endregion
    }
}
```

Ⅲ. User Data Store for .NET

- UDS for .NET 개요
- UDS for .NET 인터페이스
- UDS for .NET 구현

UDS for .NET 개요

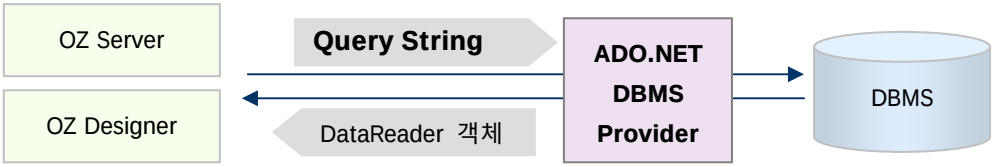
UDS(User Data Store)는 오즈가 제공하는 데이터 연동을 위한 ADO .NET Interface로 기본적으로 제공하지 않는 데이터베이스 뿐만 아니라, 데이터베이스에 저장되어 있지 않은 일반 텍스트 파일, CSV, XML 등의 파일 데이터는 물론 ASP등의 사용자 프로그램의 데이터 등 다양한 형태의 외부 데이터 소스로부터 데이터셋을 생성하여 보고서를 생성할 수 있도록 합니다.



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

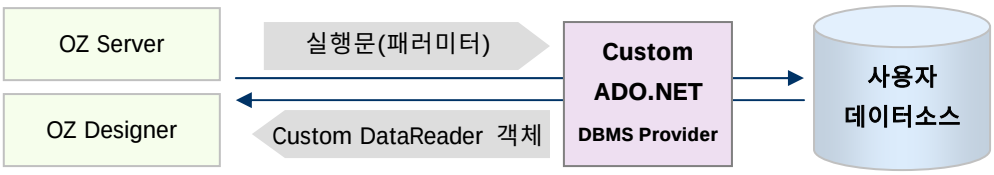
오즈는 사용자 클래스로 패러미터 스트링을 전달할 수 있으며 패러미터 값에 따라 다이나믹하게 변화하는 데이터를 전달받을 수도 있습니다.

다음은 ADO.NET과 UDS 인터페이스를 비교 설명합니다.



오즈 서버는 ADO.NET DBMS Provider를 통해 DBMS에 접속한 후 쿼리문을 전달하고 ADO.NET DBMS Provider는 DBMS로부터 전달받은 쿼리 결과 데이터를 DataReader .NET 객체에 담아 오즈 서버로 전달합니다.

이 DataReader은 ADO .NET 사양에 따르면 System.Data.IDataReader Interface를 구현(implement)한 것이어야 합니다. 즉, DBMS의 각 벤더들은 자신의 DBMS를 지원하는 ADO.NET DBMS Provider를 제공할 때 System.Data.IDataReader Interface를 구현한 Concrete DataReader 클래스를 같이 제공하는 것입니다. 이때 ADO.NET DBMS Provider를 제공하는 벤더가 반드시 지켜야 하는 것은 System.Data.IDataReader Interface를 따라 구현해야 하는 것입니다.



UDS(User Data Store)는 ADO.NET DBMS Provider의 역할을 하는 Custom ADO.NET DBMS Provider로 사용자가 System.Data.IDataReader interface를 따르면서 사용자 데이터 소스에 있는 데이터를 읽을 수 있는 DataReader 클래스를 제공해 주는 것입니다.

UDS for .NET 인터페이스

UDS 인터페이스의 클래스 구조 및 제공되는 정보는 다음과 같습니다.

OZUserDataReaderStore 인터페이스

ADO.NET DBMS Provider는 기능도 많고 복잡하여 구현하기가 힘듭니다. 그러나 오즈는 oz.uds.OZUserDataReaderStore라는 Interface를 제공하여 사용자가 OZUserDataReaderStore Interface를 따르는 클래스와 IDataReader Interface를 따르는 클래스만 작성하면 어떠한 데이터 저장소와도 연동할 수 있도록 합니다. oz.uds.OZUserDataReaderStore interface는 다음과 같이 단 4개의 메소드만을 선언하고 있습니다.

```
using System;
using System.Data;

namespace oz.uds
{
   public abstract class OZUserDataReaderStore
      : IOZUserDataStore
   {
      abstract public void Init();
      abstract public IDataReader GetDataReader(string command);
      abstract public void FreeDataReader(IDataReader idr);
      abstract public void Close();
   }
}
```

각 메소드에 대한 설명은 아래 표와 같습니다.

메소드	설명
void Init()	UDS가 처음 생성된 후 한번 호출되며, 사용자 데이터 셋에 대한 초기화 작업을 수행합니다.
IDataReader GetDataReader(string command)	오즈 서버나 디자이너로부터 Argument(실행문)을 넘겨받아 해당 실행문에 맞는 사용자 데이터를 데이터 리더 객체로 변환하여 반환합니다.

void FreeDataReader(IDataReader idr)	해당 데이터 리더의 리소스를 해제합니다.
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.

IDataReaderStore 인터페이스

IDataReaderStore 인터페이스는 클래스를 상속받지 않고 구현합니다.

```
public interface IDataReaderStore
{
    void Init();
    void Close();
    System.Data.IDataReader GetDataReader(System.String command);
    void Release(System.Data.IDataReader reader);
}
```

각 함수에 대한 설명은 아래 표와 같습니다.

함수	설명
void Init()	UDS가 처음 생성된 후 한번 호출되며, 사용자 데이터셋에 대한 초기화 작업을 수행합니다.
System.Data.IDataReader GetDataReader(System.String command)	오즈 서버나 디자이너로부터 Argument(실행문)을 넘겨받아 해당 실행문에 맞는 사용자 데이터를 데이터 리더 객체로 변환하여 반환합니다.
void Release(System.Data.IDataReader reader)	해당 데이터 리더의 리소스를 해제합니다.
void Close()	UDS를 모두 사용한 후 마지막으로 호출되며, 사용자 데이터셋에 대한 종료 작업을 수행합니다.

IDataTableStore 인터페이스

IDataTableStore 인터페이스는 클래스를 상속받지 않고 구현합니다.

```
public interface IDataTableStore
{
    void Init();
    void Close();
    System.Data.DataTable GetDataTable(System.String command);
    void Release(System.Data.DataTable table);
}
```

함수	설명
void Init()	UDS가 처음 생성된 후 한번 호출되며, 사용자 데이터셋에 대한 초기화 작업을 수행합니다.
System.Data.DataTable GetDataTable(System.String command)	오즈 서버나 디자이너로부터 Argument(실행문)을 넘겨받아 해당 실행문에 맞는 사용자 데이터를 데이터 테이블 객체로 변환하여 반환합니다.
void Release (System.Data.DataTable table)	해당 데이터 테이블의 리소스를 해제합니다.
void Close()	UDS를 모두 사용한 후 마지막으로 호출되며, 사용자 데이터셋에 대한 종료 작업을 수행합니다.

ILoggerRef 인터페이스

ILoggerRef 인터페이스는 클래스를 상속받지 않고 구현합니다.

```
public interface ILoggerRef
{
    oz.framework.log.OZLog Logger{ set; }
}
```

각 함수에 대한 설명은 아래 표와 같습니다.

함수	설명
oz.framework.log.OZLog Logger{ set; }	로그 내용을 기록합니다.

UDS for .NET 애트리뷰트

UDS 애트리뷰트에서 제공되는 정보는 다음과 같습니다.

※ 제약사항 : 애트리뷰트를 사용하여 UDS를 구현할 경우 사용되는 함수의 패러미터는 인터페이스에 정의된 함수의 패러미터와 동일할 경우에만 동작합니다.

※ 참고사항 : 애트리뷰트를 설정할 때 postfix 형태의 애트리뷰트는 Visual Studio에서 자동 인식되므로 생략 가능합니다.

UDS 애트리뷰트

■ oz.uds.DataStoreAttribute

Definition	해당 클래스가 UDS임을 나타내는 속성입니다.
Target	Class
Argument	*dataSourceType* 제공하는 데이터 타입
Sample	[oz.uds.DataStore(typeof(IDataReader))] public class UDS{ }

■ oz.uds.InitAttribute

Definition	UDS를 초기화하는 함수의 속성입니다
Target	Method
Sample	[oz.uds.Init] public void Initialize(){ }

■ oz.uds.CloseAttribute

Definition	UDS를 종료하는 함수의 속성입니다.
Target	Method
Sample	[oz.uds.Close] public void Destruct(){ }

- **oz.uds.GetDataAttribute**

Definition	데이터를 가져오는 함수의 속성입니다.
Target	Method
Sample	[oz.uds.GetData] public IDataReader RetrieveData(string command){ }

- **oz.uds.ReleaseDataAttribute**

Definition	가져온 데이터의 리소스를 해제하는 함수의 속성으로 DataStoreAttribute의 dataSourceType에 설정한 타입의 데이터를 패러미터로 전달 받습니다.
Target	Method
Sample	[oz.uds.ReleaseData] public void FinalizeData(IDataReader reader){ }

- **oz.uds.InsertAttribute**

Definition	행 삽입 데이터 액션을 수행하는 함수의 속성입니다.
Target	Method
Sample	[oz.uds.Insert] public string InsertData(string command, OZDACData[] data, string extraArgs, Hashtable parameters){ }

- **oz.uds.DeleteAttribute**

Definition	행 삭제 데이터 액션을 수행하는 함수의 속성입니다.
Target	Method
Sample	[oz.uds.Delete] public string DeleteData(string command, OZDACData[] data, string extraArgs, Hashtable parameters){ }

- **oz.uds.UpdateAttribute**

Definition	행 변경 데이터 액션을 수행하는 함수의 속성입니다.
Target	Method

Sample	[oz.uds.Update] public string UpdateAttribute(string command, OZDACData[] data1, OZDACData[] data2, string extraArgs, Hashtable parameters){ }

■ **oz.uds.CommitAttribute**

Definition	데이터 액션을 가져와서 queue에 넣고 queue를 서버로 전송하는 함수의 속성입니다.
Target	Method
Sample	[oz.uds.Commit] public string CommitChanges(){ }

■ **oz.uds.RollbackAttribute**

Definition	데이터 액션을 취소하는 함수의 속성입니다.
Target	Method
Sample	public string DropChanges(){ }

■ **oz.uds.SetHttpContextAttribute**

Definition	HttpContext를 설정하는 함수의 속성입니다.
Target	Method, Property
Sample	[oz.uds.SetHttpContext] public void SetContext(HttpContext context){ } 또는 [oz.uds.SetHttpContext] public HttpContextContext{set{_ctx = value;}}

■ **oz.uds.SetLoggerAttribute**

Definition	로그 내용을 기록하는 함수의 속성입니다.
Target	Method, Property
Sample	[oz.uds.SetLogger] public void SetLogger(oz.framework.log.OZLog log){ } 또는 [oz.uds.SetLogger] public OZLog Logger{set{log = value;}}

■ **oz.uds.SetParameterAttribute**

Definition	사용자 지정 패러미터를 설정하는 함수의 속성입니다.
Target	Method, Property
Sample	[oz.uds.SetParameter] public void SetParameter(Hashtable parameters){ } 또는 [oz.uds.SetParameter] public Hashtable{set{_parameters = value;}}

■ **oz.uds.GetAliasesAttribute**

Definition	오즈 서버의 DB 커넥션 풀 참조 시 DB 앨리어스 이름을 리스트 형태로 전달하는 함수의 속성입니다.
Target	Method, Property
Sample	[oz.uds.GetAliases] public string[] GetAliases(){ } 또는 [oz.uds.GetAliases] public string[] Aliases{get{return new string[]{"alias", ...}}

■ **oz.uds.SetConnections**

Definition	GetAliasesAttribute 속성에 정의된 DB 앨리어스의 DB 커넥션 풀을 설정하는 함수의 속성입니다.
Target	Method, Property
Sample	[oz.uds.SetConnections] public void SetConnections(IDictionary connections){ 또는 [oz.uds.SetConnections] public IDictionary Connections{set{_connections = value;}}

UDS for .NET 구현

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

UDS Source 구성

- **UDS Main 구현**

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

```
using System;
using System.Web;
using System.Data;
using System.Data.SqlClient;
using System.Collections;
using oz.uds;

namespace oz.uds
{
   public class UserDataReaderStore
      : OZUserDataReaderStore
   {
      public UserDataReaderStore()
      {
      }

      public override IDataReader GetDataReader(string command)
      {
       // Command에 따라서 사용자 데이터를 받고 IDataReader으로 변환 후 반환합니다.
       // Command는 OZ 디자이너에서 보고서 디자인 시에 넘기도록 설정합니다.
      }

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

      public override void FreeDataReader(IDataReader reader)
      {
```

```
        // GetDataReader에서 받아온 IDataReader을 모두 사용한 후 호출합니다.
        // 일반적으로 IDataReader을 Close하는 작업을 수행합니다.
      }

      public override void Init()
      {
       // UDS가 처음 생성된 후 한번 호출됩니다. DB 연결 등의 초기화 작업을 수행합니다.
      }
}
```

■ **사용자 객체 사용할 경우 DataReader 제공**

사용자 프로그램에서 DataReader 객체를 바로 넘겨 받을 수 있다면 별도의 DataReader을 구현할 필요가 없습니다. 그러나 일반적으로 사용자 애플리케이션은 String이나 ArrayList 형태로 데이터를 넘겨주기 때문에 DataReader를 직접 구현을 해줘야 하는 불편함이 있습니다. 하지만 oz에서는 oz.uds.dr에서 직접 API를 제공하기 때문에 편리하게 DataReader를 사용할 수 있습니다.

oz에서 제공되는 사용자 객체를 사용할 경우 DataReader 관련 클래스는 다음과 같습니다.

사용자 DataReader 구현 클래스
- string IDataReader 클래스 API public ArrayDataReader(string[] fieldNames, string[][] data) public ArrayDataReader(string[] fieldNames, Type[] types, string[][] data)
- ArrayList IDataReader 클래스 API public ArrayListDataReader(ArrayList fieldNames, ArrayList[] data) public ArrayListDataReader(ArrayList fieldNames, ArrayList types, ArrayList[] data)
- Dynamic DataReader 클래스 API public DynamicDataReader(IDataReader reader) * Dynamic Field인 경우 ArrayDataReader, ArrayListDataReader로 통해 호출하면 됨

UDS 예제 #1

다음은 DB로 부터 읽어서 DataReader로 처리하는 UDS 예제입니다.

```
< UserDataReaderStore.cs>

using System;
using System.Web;
using System.Data;
using System.Data.SqlClient;
using System.Collections;
```

```
using oz.uds;

namespace oz.uds
{
   public class UserDataReaderStore
      : OZUserDataReaderStore, IConnectionRef
   {
      IDbConnection con;

      public UserDataReaderStore()
      {
      }

      public override IDataReader GetDataReader(string command)
      {
         IDbCommand cmd = con.CreateCommand();
         cmd.CommandText = command;
         return cmd.ExecuteReader();
      }

      public override void Close()
      {
         con.Close();
      }

      public override void FreeDataReader(IDataReader reader)
      {
         reader.Close();
      }

      public override void Init()
      {
      }

      // oz server alias name
      private const string s_alias = "test";

      public IDictionary Connections
      { set{ con = (IDbConnection)value[s_alias]; } }

      public string[] Aliases
      { get{  return new string[]{s_alias }; } }
  }
}
```

UDS 예제 #2

다음은 DB로 부터 읽어서 DataTable로 처리하는 UDS 예제입니다.

```
< UserDataTableStore.cs>

using System;
using System.Web;
using System.Data;
using System.Collections;
using System.Data.SqlClient;

using oz.uds;

namespace oz.uds
{
   public class UserDataTableStore
      : OZUserDataTableStore, IConnectionRef
   {
      IDbConnection con;

      public UserDataTableStore()
      {
      }

      public override DataTable GetDataTable(string command)
      {
         DataTable dt = new DataTable();
         SqlDataAdapter da = new SqlDataAdapter();
         IDbCommand cmd = con.CreateCommand();
         cmd.CommandText = command;
         da.SelectCommand = (SqlCommand)cmd;
         da.Fill(dt);
         return dt;
      }

      public override void Close()
      {
         con.Close();
      }

      public override void FreeDataTable(DataTable dt)
      {
         dt.Dispose();
      }

      public override void Init()
      {
      }
```

```
      private const string s_alias = "test";

      public IDictionary Connections
      { set{ con = (IDbConnection)value[s_alias]; } }

      public string[] Aliases
      { get{  return new string[]{s_alias }; } }

   }
}
```

UDS 예제 #3

다음은 사용자 string Data로 부터 읽어서 처리하는 UDS 예제입니다.

```
< UserStringStore.cs>

using System;
using System.Web;
using System.Data;
using System.Collections;

using oz.uds;
using oz.uds.dr;

namespace oz.uds
{
  public class UserStringStore
  : OZUserDataReaderStore
  {
    string[] fieldname = {"field1", "field2"};
    string[][]  data  =  new  string[][]{new  string[]{"data11","data12"},  new
string[]{"data21", "data22"}};

    public UserStringStore()
    {
    }

    public override IDataReader GetDataReader(string command)
    {
      return new ArrayDataReader(fieldname,data);
    }

    public override void Close()
    {
    }

    public override void FreeDataReader(IDataReader reader)
```

```
    {
    }

    public override void Init()
    {
    }
  }
}
```

UDS 예제 #4

다음은 특정 디렉토리의 파일 시스템 정보를 사각형 배열로 가져오는 예제입니다.

```
< DataGenerator.cs >

using System;
using System.IO;
using System.Data;

using oz.uds;
using oz.util;

namespace oz.uds
{
   /// <summary>
   /// DataGenerator
   /// </summary>
   public class DataGenerator : OZUserDataReaderStore
   {
      public DataGenerator()
     {
     }
     public override void Init(){}

     public override void Close(){}

     public override IDataReader GetDataReader(string command)
     {
       DirectoryInfo directory = new DirectoryInfo(command);

       FileSystemInfo[] infos = directory.GetFileSystemInfos();
       /// Rectangular Array
       string[] fieldNames = {"Name", "Attributes", "Creation Time", "Last
Modified Time"};
       string[,] data = new string[infos.Length, fieldNames.Length];
       int index = 0;

       foreach(FileSystemInfo info in infos)
```

```
        {
            data[index,0] = info.Name;
            data[index,1] = info.Attributes.ToString();
            data[index,2] = info.CreationTime.ToString();
            data[index,3] = info.LastWriteTime.ToString();
            index++;
          }
          return new oz.uds.dr.ArrayDataReader(fieldNames, data);
       }
       public override void FreeDataReader(IDataReader idr){}
   }
}
```

UDS 예제 #5

다음은 클래스를 상속받지 않고 DB로 부터 읽어서 IDataReaderStore로 처리하는 UDS 예제입니다.

```
using System;
using System.Web;
using System.Data;
using System.Collections;
using oz.uds;
using oz.uds.dr;

namespace oz.uds.sample.csharp.@interface{
   public class ArrayDataReaderSample : IDataReaderStore, IHttpContextRef,
IParameterRef
   {
       private HttpContext _ctx;
       private IDictionary _params;
       public HttpContext HttpContext{ set{ _ctx = value; } }
       public IDictionary Parameters{ set{ _params = value; } }
     public void Init()
       {
       }
       public IDataReader GetDataReader(string command)
       {
           int                          fieldCount                          =
oz.util.OZString.ParseInt(Convert.ToString(_params["field_count"]), 5);
           int                           rowCount                           =
oz.util.OZString.ParseInt(Convert.ToString(_params["row_count"]), 5);
           bool                        useJaggedArray                        =
oz.util.OZString.ParseBool(Convert.ToString(_params["use_jagged_array"]),
false);
           string[] fieldNames = new string[fieldCount];
           string[][] jagged = null;
           string[,] rectangular = null;
```

```
            if(useJaggedArray)
                jagged = new string[rowCount][];
            else
                rectangular = new string[rowCount, fieldCount];
            for(int col = 0; col < fieldCount; col++)
                fieldNames[col] = "Field" + col;

            for(int row = 0; row < rowCount; row++)
            {
                if(useJaggedArray)
                    jagged[row] = new string[fieldCount];
                for(int col = 0; col < fieldCount; col++)
                {
                    string value = "Value - " + row + "," + col;
                    if(useJaggedArray)
                        jagged[row][col] = value;
                    else
                        rectangular[row,col] = value;
                }
            }
            if(useJaggedArray)
                return new ArrayDataReader(fieldNames, jagged);
            else
                return new ArrayDataReader(fieldNames, rectangular);
        }
        public void Release(IDataReader reader)
        {
        }
        public void Close()
        {
        }
    }
}
```

UDS 예제 #6

다음은 클래스를 상속받지 않고 데이터 테이블로 부터 읽어서 IDataTableStore로 처리하는 UDS 예제입니다.

```
using System;
using System.Web;
using System.Data;
using System.Collections;

using oz.uds;
using oz.uds.dr;
using oz.framework.dac;
```

```
namespace oz.uds.sample.csharp.@interface
{
   public class UserDataTableStoreSample : IDataTableStore, IParameterRef,
ILoggerRef, IDataAction
   {
      private HttpContext _ctx;
      private IDictionary _params;
      private oz.framework.log.OZLog log;

      public HttpContext HttpContext{ set{ _ctx = value; } }

      public IDictionary Parameters{ set{ _params = value; } }

      public oz.framework.log.OZLog Logger{ set{ log = value; } }

      public void Init()
      {
      }

      public DataTable GetDataTable(string command)
      {
         int fieldCount =
oz.util.OZString.ParseInt(Convert.ToString(_params["field_count"]), 5);
         int rowCount =
oz.util.OZString.ParseInt(Convert.ToString(_params["row_count"]), 5);

         DataTable table = new DataTable();
         for(int i = 0; i < fieldCount; i++)
            table.Columns.Add("field" + i);

         for(int rowIndex = 0; rowIndex < rowCount; rowIndex++)
         {
            DataRow row = table.NewRow();
            for(int colIndex = 0; colIndex < fieldCount; colIndex++)
            {
               row[colIndex] = "value " + rowIndex + ", " + colIndex;
            }

            table.Rows.Add(row);
         }

         return table;
      }

      public void Release(DataTable table)
      {
         if(null != table)
            table.Dispose();
      }
```

```
        public void Close()
        {
        }



        public void Rollback()
        {
            log.Debug("Rolling back. ");
        }

        public string Commit()
        {
            log.Debug("Committing. ");
            return "commit";
        }

        public string UpdateRow(string command, oz.framework.dac.OZDACData[]
condition, oz.framework.dac.OZDACData[] source, string extra, Hashtable
parameters)
        {
            log.Debug("Update command : " + command);

            foreach(OZDACData data in condition)
                log.Debug(data.FieldName + "=" + data.FieldData);

            foreach(OZDACData data in source)
                log.Debug(data.FieldName + "=" + data.FieldData);

            foreach(DictionaryEntry entry in parameters)
            {
                log.Debug(entry.Key + "=" + entry.Value);
            }

            return "update";
        }

        public string InsertRow(string command, oz.framework.dac.OZDACData[]
source, string extra, Hashtable parameters)
        {
            log.Debug("Update command : " + command);

            foreach(OZDACData data in source)
                log.Debug(data.FieldName + "=" + data.FieldData);

            foreach(DictionaryEntry entry in parameters)
            {
                log.Debug(entry.Key + "=" + entry.Value);
            }
```

```
            return "insert";
        }

        public  string  DeleteRow(string  command,  oz.framework.dac.OZDACData[]
condition, string extra, Hashtable parameters)
        {
            log.Debug("Update command : " + command);

            foreach(OZDACData data in condition)
                log.Debug(data.FieldName + "=" + data.FieldData);

            foreach(DictionaryEntry entry in parameters)
            {
                log.Debug(entry.Key + "=" + entry.Value);
            }

            return "delete";
        }
    }
}
```

UDS 예제 #7

다음은 클래스를 상속받지 않고 DB로 부터 읽어서 애트리뷰트로 처리하는 UDS 예제입니다.

```
using System;
using System.Web;
using System.Data;
using System.Collections;

using oz.uds;
using oz.uds.dr;

namespace oz.uds.sample.csharp.attribute
{
    [DataStore(typeof(ArrayDataReader))]
    public class ArrayDataReaderSample
    {
        private oz.framework.log.OZLog log;

        private HttpContext _ctx;
        private IDictionary _params;

        [SetHttpContext]
        public HttpContext HttpContext{ set{ _ctx = value; } }

        [SetParameter]
        public IDictionary Parameters{ set{ _params = value; } }
```

```
      [SetLogger]
      public oz.framework.log.OZLog Logger{ set{ log = value; } }

      [Init]
      public void Init()
      {
      }

      [GetData]
      public IDataReader GetDataReader(string command)
      {
         log.Debug("Creating result set. " + command);
         int                            fieldCount                            =
oz.util.OZString.ParseInt(Convert.ToString(_params["field_count"]), 5);
         int                             rowCount                             =
oz.util.OZString.ParseInt(Convert.ToString(_params["row_count"]), 5);

         bool                          useJaggedArray                         =
oz.util.OZString.ParseBool(Convert.ToString(_params["use_jagged_array"]),
false);

         string[] fieldNames = new string[fieldCount];
         string[][] jagged = null;
         string[,] rectangular = null;
         if(useJaggedArray)
            jagged = new string[rowCount][];
         else
            rectangular = new string[rowCount, fieldCount];

         for(int col = 0; col < fieldCount; col++)
            fieldNames[col] = "Field" + col;

         for(int row = 0; row < rowCount; row++)
         {
            if(useJaggedArray)
               jagged[row] = new string[fieldCount];

            for(int col = 0; col < fieldCount; col++)
            {
               string value = "Value - " + row + "," + col;
               if(useJaggedArray)
                  jagged[row][col] = value;
               else
                  rectangular[row,col] = value;
            }
         }

         if(useJaggedArray)
            return new ArrayDataReader(fieldNames, jagged);
```

```
            else
                return new ArrayDataReader(fieldNames, rectangular);
        }

        [ReleaseData]
        public void FreeDataReader(IDataReader reader)
        {
        }

        [Close]
        public void Close()
        {
        }
    }
}
```

UDS 예제 #8

다음은 클래스를 상속받지 않고 데이터 테이블로 부터 읽어서 애트리뷰트로 처리하는 UDS 예제입니다.

```
using System;
using System.Web;
using System.Data;
using System.Collections;

using oz.uds;
using oz.uds.dr;
using oz.framework.dac;

namespace oz.uds.sample.csharp.attribute
{
    [DataStore(typeof(DataTable))]
    public class UserDataTableStoreSample
    {
        private oz.framework.log.OZLog log;

        private HttpContext _ctx;
        private IDictionary _params;

        [SetHttpContext]
        public HttpContext HttpContext{ set{ _ctx = value; } }

        [SetParameter]
        public IDictionary Parameters{ set{ _params = value; } }

        [SetLogger]
        public oz.framework.log.OZLog Logger{ set{ log = value; } }
```

```
      [Init]
      public void Init()
      {
      }

      [GetData]
      public DataTable GetDataTable(string command)
      {
         int                              fieldCount                              =
oz.util.OZString.ParseInt(Convert.ToString(_params["field_count"]), 5);
         int                               rowCount                               =
oz.util.OZString.ParseInt(Convert.ToString(_params["row_count"]), 5);

         DataTable table = new DataTable();
         for(int i = 0; i < fieldCount; i++)
            table.Columns.Add("field" + i);

         for(int rowIndex = 0; rowIndex < rowCount; rowIndex++)
         {
            DataRow row = table.NewRow();
            for(int colIndex = 0; colIndex < fieldCount; colIndex++)
            {
               row[colIndex] = "value " + rowIndex + ", " + colIndex;
            }

            table.Rows.Add(row);
         }

         return table;
      }

      [ReleaseData]
      public void Release(DataTable table)
      {
         if(null != table)
            table.Dispose();
      }

      [Close]
      public void Close()
      {
      }

      [Rollback]
      public void DropChanges()
      {
         log.Debug("Rolling back. ");
      }
```

```
      [Commit]
      public string Mirror()
      {
         log.Debug("Committing. ");
         return "commit";
      }

      [Update]
      public string ModifyData(string command, oz.framework.dac.OZDACData[]
condition, oz.framework.dac.OZDACData[] source, string extra, Hashtable
parameters)
      {
         log.Debug("Update command : " + command);

         foreach(OZDACData data in condition)
            log.Debug(data.FieldName + "=" + data.FieldData);

         foreach(OZDACData data in source)
            log.Debug(data.FieldName + "=" + data.FieldData);

         foreach(DictionaryEntry entry in parameters)
         {
            log.Debug(entry.Key + "=" + entry.Value);
         }

         return "update";
      }

      [Insert]
      public string PutData(string command, oz.framework.dac.OZDACData[] source,
string extra, Hashtable parameters)
      {
         log.Debug("Update command : " + command);

         foreach(OZDACData data in source)
            log.Debug(data.FieldName + "=" + data.FieldData);

         foreach(DictionaryEntry entry in parameters)
         {
            log.Debug(entry.Key + "=" + entry.Value);
         }

         return "insert";
      }

      [Delete]
      public string RemoveData(string command, oz.framework.dac.OZDACData[]
condition, string extra, Hashtable parameters)
      {
         log.Debug("Update command : " + command);
```

```
            foreach(OZDACData data in condition)
                log.Debug(data.FieldName + "=" + data.FieldData);

            foreach(DictionaryEntry entry in parameters)
            {
                log.Debug(entry.Key + "=" + entry.Value);
            }

            return "delete";
        }
    }
}
```

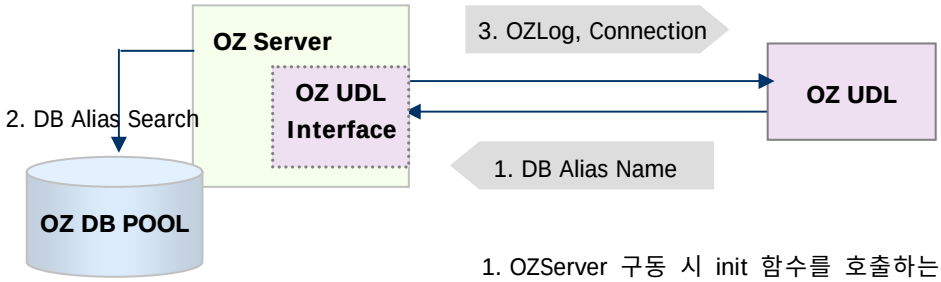
Ⅳ. User Defined Log

- UDL 개요
- UDL 인터페이스
- UDL 구현

UDL 개요

UDL(User Defined Log)은 오즈 서버를 통하여 데이터 액션을 수행하거나 데이터를 조회할 경우 데이터 베이스나 보고서 관련 정보를 임의의 파일이나 DB에 사용자가 정의한 타입의 형식으로 기록할 수 있도록 하기 위한 인터페이스입니다.

UDL 인터페이스의 호출 구조는 아래 그림과 같습니다.



1. OZServer 구동 시 init 함수를 호출하는 경우
2. 서비스 호출하여 traceLog하는 경우

UDL 인터페이스

UDL 인터페이스의 클래스 구조 및 제공되는 정보는 다음과 같습니다.

OZUserDefinedLogTarget 인터페이스의 클래스 구조

OZ UDL에 제공되는 정보

Target 클래스	정보	설명
IUserDefinedLog Target	IP Address	클라이언트가 요청한 오즈 서버 IP 주소
	HttpRequest	클라이언트에서 요청한 HttpRequest
	HttpContext	클라이언트에서 요청한 HttpContext
	UserID	클라이언트에서 쿼리문을 실행한 오즈 서버 사용자 ID
IDMLogTarget extends IUserDefinedLog Target	ODIName	쿼리문이 실행된 ODI 이름
	DataSetName	쿼리문이 실행된 데이터 셋 이름
	DataStoreName	쿼리문이 실행된 데이터 스토어 이름
	OZParam	오즈 사용자 지정 패러미터를 Hash 형태로 제공
IRDBLogTarget extends IDMLogTarget	DBAlias	실행한 ODI에서 사용한 커넥션 풀의 DB 앨리어스 명 ※ 참고사항 : 데이터 스토어에서 앨리어스를 사용하지 않는 경우 커넥션 풀의 키 값을 넘겨 줍니다.
	ExecuteQuery	요청한 쿼리문

	PreparedQueryValue	컴파일된 질의문을 사용할 경우 사용된 값
	QueryState	쿼리문이 정상적으로 실행되었는지 여부
	QueryExecuteTime	쿼리 수행 시간
	isPrepared	컴파일된 질의문 사용 여부
	CRUDFlagName	실행된 쿼리문의 타입 (select/insert/update/delete)
IMonitorLogTarget extends IUserDefinedLogTarget	MARK	서비스 시작 및 종료를 표시하는 문자열 (start/end)
	ThreadID	서비스 스레드 ID
	Time	서비스 시작 및 종료 시간 (단위 : ms) ※ 참고사항 : 1970년 1월 1일부터 현재 시간을 Millisecond로 표시 합니다.
	Free Memory	사용 가능한 메모리 (단위 : byte)
	Total Memory	전체 메모리 (단위 : byte)
	Service Code	서비스 구분 코드 ※ 참고사항 - MARK가 "end"일 경우에만 값이 리턴되며 "start"일 경우 -1을 리턴합니다. - 자세한 서비스 구분 코드는 오즈 엔터 프라이즈 서버 관리자 가이드의 "IV. 오즈 엔터프라이즈 서버 설정" 부분을 참조하시기 바랍니다.
	Service Status	서비스 상태 코드 ※ 참고사항 - MARK가 "end"일 경우에만 값이 리턴되며 "start"일 경우 -1을 리턴합니다. - 성공일 경우 "9001", 실패일 경우 "9002"를 리턴합니다.
	Service Param	서비스 패러미터 ※ 참고사항 : MARK가 "end"일 경우에만 값이 리턴되며 "start"일 경우 -1을 리턴합니다.
	DB SessionID	DBMS 세션 ID ※ 참고사항 : MARK가 "end"일 경우에만 값이 리턴되며 "start"일 경우 -1을 리턴합니다.
	Execute Time	처리 시간 ※ 참고사항 : MARK가 "end"일 경우에만 값이 리턴되며 "start"일 경우 -1을 리턴합니다.

	MARK	서비스 시작 및 종료를 표시하는 문자열 (start/end)

OZ UDL 인터페이스

■ **oz.udl.OZUserDefinedLogger 클래스**

OZ UDL을 사용하기 위하여 oz.udl.OZUserDefinedLogger를 구현한 후 init, traceLog, getOZDBAlias 메소드를 반드시 구현하여야 하며 해당 메소드는 인터페이스를 정의하는 방법 외에 애트리뷰트를 이용하여 지정할 수 있습니다.

- init

Prototype	`public void init(oz.framework.db.OZConnection ozconn, com.forcs.log4oz.OZLog log)`	
Definition	오즈 서버 구동 시 사용자 정의 로그를 남길 대상을 정의합니다.	
Argument	*ozconn*	DB에 사용자 정의 로그를 기록할 경우 사용할 DB 커넥션 정보 및 기록할 로그 형식
	log	파일에 사용자 정의 로그를 기록할 경우 사용할 파일 정보 및 기록할 로그 형식

- Trace

Prototype	`public void Trace(oz.udl.IUserDefinedLogTarget target, oz.framework.db.OZConnection ozconn, oz.framework.log.OZLog log)`	
Definition	로그에 기록할 타겟 내용을 정의합니다.	
Argument	*target*	사용자 정의 로그 타겟
	ozconn	DB에 사용자 정의 로그를 기록할 경우 사용할 DB 커넥션 정보 및 기록할 로그 형식
	log	파일에 사용자 정의 로그를 기록할 경우 사용할 파일 정보 및 기록할 로그 형식

- getOZDBAlias

Prototype	`public java.lang.String getOZDBAlias() throws OZUserDefinedLogException`
Definition	DB에 사용자 정의 로그를 기록할 경우 오즈 서버 커넥션 풀의 앨리어스 명 중 로그를 기록할 앨리어스 명을 설정합니다.
return	DB 앨리어스 명

※ 참고사항 : 애트리뷰트를 이용하여 구현하는 방법

애트리뷰트를 이용하여 구현할 경우에는 대괄호를 이용하여 애트리뷰트 명을 지정(ex. [OSUDLInitialize])하고 그 다음 줄에 함수를 선언하여 구현합니다.

만일 다음과 같이 구현된 메소드를

```
public class UserDefinedLogger : IUserDefinedLogger{
public void Init(OZConnection con, OZLog log){}
public void Trace(IUserDefinedLogTarget target, OZConnection
con, OZLog log){}

public string DbAlias{ get; }
}
```

아래처럼 애트리뷰트를 이용하여 구현할 수 있습니다.

```
public class UserDefinedLogger{
[OZUDLInitialize]
public void Init(OZConnection con, OZLog log){}

[OZUDLTrace]
public void Trace(IUserDefinedLogTarget target, OZConnection
con, OZLog log){}

[OZUDLDbAlias]
public string DbAlias{ get; }
}
```

■ IUserDefinedLogTarget 클래스

OZ UDL에서 제공되는 정보를 요청하여 사용하기 위한 인터페이스입니다.

- IPAddress

Prototype	`public String IPAddress(){get;}`
Definition	클라이언트가 요청한 오즈 서버 IP 주소를 가져옵니다
return	오즈 서버 IP 주소

- HttpRequest

Prototype	`public HttpServletRequest HttpRequest(){get;}`
Definition	클라이언트에서 요청한 HttpRequest를 가져옵니다.
return	HttpRequest

- HttpContext

Prototype	`public HttpServlet HttpContext(){get;}`
Definition	클라이언트에서 HttpContext를 가져옵니다.
return	HttpContext

- UserID

Prototype	`public String UserID(){get;}`
Definition	클라이언트에서 쿼리문을 실행한 오즈 서버 사용자 ID를 가져옵니다
return	사용자 ID

■ **IDMLogTarget 클래스**

OZ UDL에서 데이터 모듈 관련 정보를 사용하기 위한 인터페이스입니다.

- ODIName

Prototype	`public String ODIName(){get;}`
Definition	쿼리문을 실행한 ODI 이름을 가져옵니다.
return	ODI 이름

- DataSetName

Prototype	`public String DataSetName(){get;}`
Definition	쿼리문을 실행한 데이터 셋 이름을 가져옵니다.
return	데이터 셋 이름

- DataStoreName

Prototype	`public String DataStoreName(){get;}`
Definition	쿼리문을 실행한 데이터 스토어 이름을 가져옵니다.
return	데이터 스토어 이름

- Parameters

Prototype	`public System.Connection.IDictionary Parameters{get;}`
Definition	오즈 사용자 지정 패러미터를 Dictionary 형태로 가져옵니다.
return	오즈 사용자 지정 패러미터의 key와 Value

■ **IRDBLogTarget 클래스**

OZ UDL에서 RDB 스토어 관련 정보를 사용하기 위한 인터페이스입니다.

- DBAlias

Prototype	`public String DBAlias{get;}`
Definition	실행한 ODI에서 사용한 커넥션 풀의 DB 앨리어스 명을 가져옵니다.
return	DB 앨리어스 명 ※ 참고사항 : 데이터 스토어에서 앨리어스를 사용하지 않는 경우 커넥션 풀의 키 값을 가져옵니다.

- Query

Prototype	`public String Query{get;}`
Definition	클라이언트에서 요청한 쿼리문을 가져옵니다.
return	쿼리문

- PreparedQueryValue

Prototype	`public Vector PreparedQueryValue{get;}`
Definition	컴파일된 질의문을 사용할 경우 사용된 값을 가져옵니다.
return	컴파일된 질의문에 사용된 값 (바이너리 데이터 포함)

- QueryState

Prototype	`public String QueryState{get;}`	
Definition	클라이언트에서 요청한 쿼리문이 정상적으로 실행되었는지 여부를 가져옵니다.	
return	true	쿼리문이 정상적으로 실행됨
	flase	쿼리문이 정상적으로 실행되지 않음

- QueryExecuteTime

Prototype	`public long QueryExecuteTime{get;}`
Definition	클라이언트에서 요청한 쿼리문이 수행되는 시간을 가져옵니다.
return	쿼리 수행 시간

- isPrepared

Prototype	`public String isPrepared{get;}`	
Definition	컴파일된 질의문의 사용 여부를 가져옵니다.	
return	true	컴파일된 질의문을 사용함
	flase	컴파일된 질의문을 사용하지 않음

- QueryType

Prototype	`public String QueryType{get;}`
Definition	클라이언트에서 요청하여 실행된 쿼리문의 타입을 가져옵니다.
return	select Select 문이 사용됨 insert Insert 문이 사용됨 update Update 문이 사용됨 delete Delete 문이 사용됨

■ **IMLogTarget 클래스**

OZ UDL에서 모니터 로그 관련 정보를 사용하기 위한인터페이스입니다.

- ODIName

Prototype	`public String ODIName(){get;}`
Definition	쿼리문을 실행한 ODI 이름을 가져옵니다.
return	ODI 이름

- Mark

Prototype	`public string Mark(){get;}`
Definition	서비스 시작 및 종료 표시를 가져옵니다.
return	start 서비스 시작 end 서비스 종료

- ThreadID

Prototype	`public integer ThreadID(){get;}`
Definition	서비스 스레드 ID를 가져옵니다.
return	스레드 ID

- ServiceTime

Prototype	`public long getServiceTime(){get;}`
Definition	서비스 시작 시간과 종료 시간을 가져옵니다. (단위 : ms) ※ 참고사항 : 1970년 1월 1일부터 현재 시간을 Millisecond로 표시합니다.
return	서비스 시작 시간 및 종료 시간

- FreeMemory

Prototype	`public long FreeMemory(){get;}`

Definition	사용 가능한 메모리 크기를 가져옵니다. (단위 : byte)
return	메모리 크기

- TotalMemory

Prototype	`public long TotalMemory(){get;}`
Definition	전체 JVM 메모리 크기를 가져옵니다. (단위 : byte)
return	메모리 크기

- ServiceCode

Prototype	`public integer ServiceCode(){get;}`
Definition	서비스 구분 코드를 가져옵니다. ※ 참고사항 - MARK가 "end"일 경우에만 값이 리턴되며 "start"일 경우 -1을 리턴합니다. - 자세한 서비스 구분 코드는 오즈 엔터 프라이즈 서버 관리자 가이드의 "IV. 오즈 엔터프라이즈 서버 설정" 부분을 참조하시기 바랍니다.
return	서비스 구분 코드

- ServiceStatus

Prototype	`public integer ServiceStatus(){get;}`	
Definition	서비스 상태 코드를 가져옵니다. ※ 참고사항 : MARK가 "end"일 경우에만 값이 리턴되며 "start"일 경우 -1을 리턴합니다.	
return	9001	서비스 성공
	9002	서비스 실패

- ServiceParameter

Prototype	`public string ServiceParameter(){get;}`
Definition	클라이언트에서 서비스 요청 시 전송되는 패러미터 값을 가져옵니다. ※ 참고사항 : MARK가 "end"일 경우에만 값이 리턴되며 "start"일 경우 -1을 리턴합니다.
return	패러미터 값

- DBSessionID

Prototype	`public string DBSessionID(){get;}`
Definition	DBMS 세션 ID를 가져옵니다. ※ 참고사항 : MARK가 "end"일 경우에만 값이 리턴되며 "start"일 경우 -1을 리턴합니다.

return	DBMS 세션 ID

- ExecuteTime

Prototype	`public string ExecuteTime(){get;}`
Definition	서비스 처리 시 소요된 시간을 가져옵니다. (단위 : ms) ※ 참고사항 : MARK가 "end"일 경우에만 값이 리턴되며 "start"일 경우 -1을 리턴합니다.
return	서비스 처리 시간

UDL 구현

서버 로그와 모니터 로그에 UDL을 적용하여 사용자 정의 로그를 파일이나 데이터 베이스에 기록하기 위한 방법에 대해 설명합니다.

오즈 서버 설정

■ **관련 파일**

- UDL 관련 라이브러리 파일 : OZ_HOME/bin/ozudl.dll
- Lof4j 관련 라이브러리 파일 : OZ_HOME/bin/log4net.dll
- UDL 관련 설정 파일 : OZ_HOME/conf/ozudl.properties

■ **파일 설정값 변경**

- udlmngr.properties 파일을 열어 UDL 사용 여부를 설정하고 사용할 경우 구현한 클래스의 경로를 설정합니다.

```
#---------------------------------
# configuraion of OZ User Defined log
#---------------------------------

OZ_USER_DEFINED_LOG.Active=true
OZ_USER_DEFINED_LOG.Class=oz.udl.UDL클래스명

OZ_UDL_MONITOR.Active= true
OZ_UDL_MONITOR.Class=oz.udl.UDL클래스명
```

  - ozudl.properties에 아래와 같이 설정하여 UDL을 사용함으로 설정합니다.
    - 서버 로그를 사용자 정의로그로 기록할 경우
      OZ_USER_DEFINED_LOG.Active=true
    - 모니터 로그를 사용자 정의로그로 기록할 경우
      OZ_UDL_MONITOR.Active= true

  - 서버로그와 모니터 로그의 UDL 클래스를 설정합니다.
    - OZ_USER_DEFINED_LOG.Class=oz.udl.UDL클래스명
    - OZ_UDL_MONITOR.Class=oz.udl.UDL클래스명

      ex)

OZ_USER_DEFINED_LOG.Class=oz.udl.file.OZUserDefinedLogger

OZ_UDL_MONITOR.Class=oz.udl.db.OZMonitorLogForDB

- OZ_HOME/Web.config 파일을 열어 .Net Framework 버전을 설정하고 사용자 정의 로그를 기록할 파일의 경로를 설정합니다.

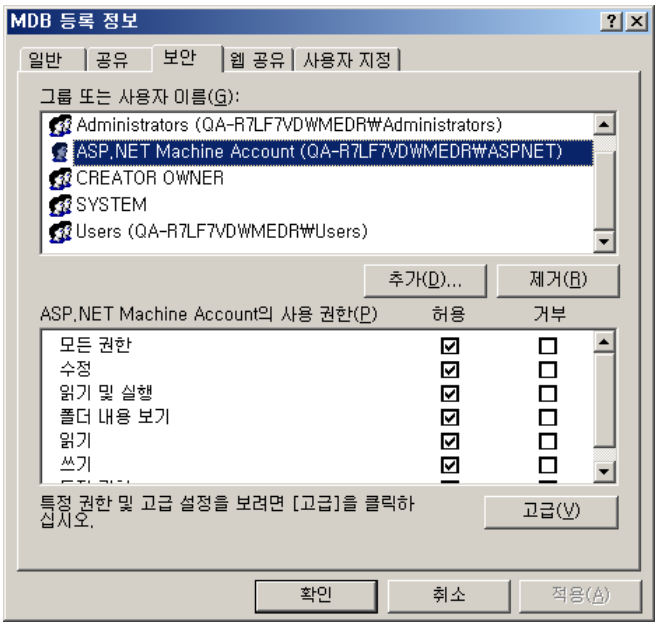
```
<?xml version="1.0" encoding="utf-8" ?>
      <configuration>
      <configSections>
<!--.Net Framework 2.0 버전 이하를 사용할 경우 반드시 아래와 같이 .Net Framework
버전을 명시하여야합니다.-->
              <sectionGroup name="udl">
                      <section name="file"
type="System.Configuration.NameValueSectionHandler, System,
Version=1.0.5000.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"/>
                      <section name="db"
type="System.Configuration.NameValueSectionHandler, System,
Version=1.0.5000.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"/>
              </sectionGroup>
      </configSections>
  <system.web>
<httpRuntime maxRequestLength="100000"  />
 </system.web>
      <udl>
              <file>
                      <add key="path" value="C:/Program Files/Forcs/50/OZ
Server.NET 5.0/logs/UDL.Log" />
                      <add key="layout" value="%d{yyyy-MM-dd HH:mm:ss.fff}
[%-5p] %C{1}.%M - %m%n" />
                      <add key="daily" value="false" />
                      <add key="size" value="100KB" />
                      <add key="maxBackup" value="5" />
                      <!--add key="daily" value="true" />
                      <add key="pattern" value="yyyy-MM-dd" /-->
              </file>
              <db>
                      <add key="alias" value="qa_ms" />
              </db>
      </udl>
</configuration>
```

※ 주의사항

- <configSections> 앨리먼트는 반드시 최상위에 위치하여야 합니다.
- .Net Framework 2.0 버전 이하를 사용할 경우 sectionGroup에 위와 같이 .Net Framework 버전을 명시하여야합니다.
- 닷넷 서버에서 ODI에서 ODBC 데이터 베이스 벤더를 사용하여 데이터 액션을 실행할 경우 해당 ODBC에서 사용하는 MDB 파일이 존재하는 폴더에 아래와 같

이 권한을 추가하여야 합니다.

MDB 파일이 존재하는 폴더에서 마우스 오른쪽 버튼을 클릭하여 나타나는 팝업 메뉴 중 [등록 정보] 메뉴를 선택합니다. 등록 정보 화면에서 [보안] 탭으로 이동하여 사용자에 ASP.NET을 추가합니다



ASP.NET 사용자의 권한 설정 시 모든 권한이 체크되어있어야 데이터 액션이 정상적으로 실행됩니다

UDL 예제 1 - 서버 로그를 파일에 기록하는 방법

UDL을 이용하여 오즈 서버 로그를 임의의 파일에 사용자가 정의한 타입의 형식으로 기록하는 방법에 대해 설명합니다.

■ **사용자 정의 로그 구현**

UDL을 사용하여 사용자 로그 파일을 생성하는 클래스 파일을 만듭니다. 본 예제에서는 아래와 같은 java 소스 파일을 이용하여 oz.udl.file.OZUserDefinedLoggerForFile 클래스를 생성한 후 해당 클래스를 "ozudl.jar" 파일로 만들었습니다.

```
#region Using Statements
using System;
using System.Web;
using System.Text;
using System.Globalization;
using System.Configuration;
using System.Collections.Specialized;
```

```
using oz.udl;

using log4net;
using log4net.Appender;
#endregion

namespace oz.udl.file
{
   public class OZUserDefinedLogger : oz.udl.IUserDefinedLogger
   {
      private string _name = "File UDL";
      private ILog log;

      public OZUserDefinedLogger(){}

      public void Init(oz.framework.db.OZConnection con, oz.framework.log.OZLog
log)
      {
         log.Debug(_name, "Called init");

         NameValueCollection                    values                    =
(NameValueCollection)ConfigurationSettings.GetConfig("udl/file");
         if(null == values)
            throw new Exception("Cannot find udl file configuration");

         // configure appender programmatically
         RollingFileAppender appender = new RollingFileAppender();
         log4net.Layout.PatternLayout          layout          =          new
log4net.Layout.PatternLayout();
         layout.ConversionPattern = values["layout"];
         layout.ActivateOptions();
         appender.Layout = layout;

         appender.File = values["path"];

         appender.AppendToFile = true;
         appender.MaximumFileSize = values["size"];
         appender.ImmediateFlush = true;

         if(0 == string.Compare(values["type"], "daily", true))
         {
            appender.RollingStyle = RollingFileAppender.RollingMode.Date;
            appender.DatePattern = values["pattern"];
         }
         else
         {
            appender.RollingStyle = RollingFileAppender.RollingMode.Size;
            appender.MaxSizeRollBackups = int.Parse(values["maxBackup"]);
         }
```

```
          appender.ActivateOptions();
          this.log = LogManager.GetLogger(_name);
          log4net.Repository.Hierarchy.Logger              core              =
(log4net.Repository.Hierarchy.Logger)this.log.Logger;


((log4net.Repository.IBasicRepositoryConfigurator)core.Repository).Configure(ap
pender);
     }

       public string DbAlias
       { get{ return null; } }

       public          void          Trace(IUserDefinedLogTarget          target,
oz.framework.db.OZConnection con, oz.framework.log.OZLog log)
       {
          if(log.IsDebugEnabled)
             log.Debug(_name, "Called trace");

          StringBuilder sb = new StringBuilder(4096);

          IRDBLogTarget rdbStore = target as IRDBLogTarget;

          if(null != rdbStore)
          {
             sb.AppendFormat("[DB        Alias={0}][IP        Address={1}][User
ID={2}][QueryType={3}][Query={4}][IsPrepared={5}]",
                 rdbStore.Alias,       rdbStore.IPAddress,       rdbStore.UserID,
rdbStore.QueryType, rdbStore.Query, rdbStore.IsPrepared);

             if(rdbStore.IsPrepared)
             {
                sb.Append("[PreparedValues=");
                foreach(object o in rdbStore.PreparedQueryValues)
                {
                   sb.Append(null == o ? "null" : o is byte[] ? "binary" :
o.ToString());
                   sb.Append(";");
                }
                sb.Append("]");
             }

             sb.AppendFormat("[QueryExeTime={0}][QueryState={1}]",
rdbStore.QueryExecutionTime, rdbStore.QueryState);
          }

          IDMLogTarget dataModule = target as IDMLogTarget;

          if(null != dataModule)
          {
```

```
sb.AppendFormat("[ODIName={0}][DataSetName={1}][DataStoreName={2}]",
                dataModule.ODIName,                    dataModule.DataSetName,
dataModule.DataStoreName);
          }

          NameValueCollection  parameters  =  null  !=  target.HttpRequest  ?
target.HttpRequest.Params : null;
          if(null != parameters)
          {
              sb.Append("[RequestParams=");

              foreach(string key in parameters.AllKeys)
              {
                  sb.AppendFormat("({0},{1})", key, parameters[key]);
              }
              sb.Append("]");
          }
          System.Web.SessionState.HttpSessionState   session   =   null   !=
target.HttpContext ? target.HttpContext.Session : null;
          if(null != session)
          {
              sb.Append("[SessionAttrs=");
              foreach(string key in session.Keys)
              {
                  sb.AppendFormat("({0},{1})", key, session[key]);
              }
          }

          this.log.Debug(sb.ToString());
      }
   }

   public class OZUserDefinedLogger2
   {
       private OZUserDefinedLogger _logger;

       public OZUserDefinedLogger2()
       {
          _logger = new OZUserDefinedLogger();
       }

       [OZUDLInitialize]
       public void Init(oz.framework.db.OZConnection con, oz.framework.log.OZLog
log)
       {
          _logger.Init(con, log);
       }

       [OZUDLTrace]
```

```
      public          void          Trace(IUserDefinedLogTarget          target,
oz.framework.db.OZConnection con, oz.framework.log.OZLog log)
      {
         _logger.Trace(target, con, log);
      }
   }
}
```

■ **서버 환경 설정**

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

```
#--------------------------------
# configuraion of OZ User Defined log
#--------------------------------

OZ_USER_DEFINED_LOG.Active=true
OZ_USER_DEFINED_LOG.Class=oz.udl.file.OZUserDefinedLoggerForFile
```

- 닷넷 서버가 설치된 경로의 Web.config 파일을 텍스트 편집기로 열어 파일이 생성될 경로와 파일에 기록할 로그 형식을 아래와 같이 설정한 후 저장합니다.

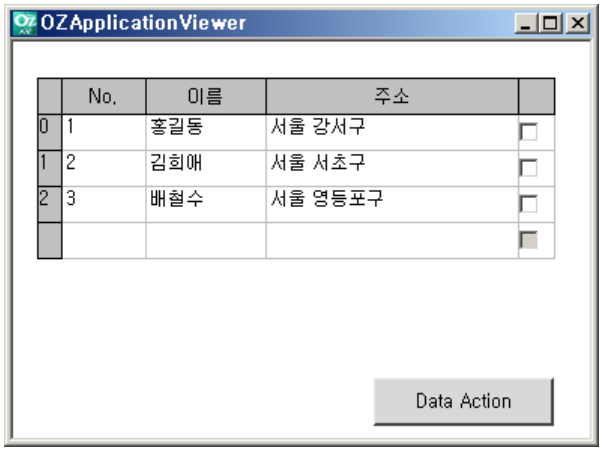
```
<?xml version="1.0" encoding="utf-8" ?>
    <configuration>
    <configSections>
<!--.Net Framework 2.0 버전 이하를 사용할 경우 반드시 아래와 같이 .Net Framework
버전을 명시하여야합니다.-->
            <sectionGroup name="udl">
                    <section name="file"
type="System.Configuration.NameValueSectionHandler, System,
Version=1.0.5000.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"/>
                    <section name="db"
type="System.Configuration.NameValueSectionHandler, System,
Version=1.0.5000.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"/>
            </sectionGroup>
    </configSections>
  <system.web>
<httpRuntime maxRequestLength="100000"  />
 </system.web>
    <udl>
            <file>
                    <add key="path" value="C:/Program Files/Forcs/50/OZ
Server.NET 5.0/logs/UDL.Log" />
                    <add key="layout" value="%d{yyyy-MM-dd HH:mm:ss.fff}
[%-5p] %C{1}.%M - %m%n" />
                    <add key="daily" value="false" />
                    <add key="size" value="100KB" />
                    <add key="maxBackup" value="5" />
```

```
                    <!--add key="daily" value="true" />
                    <add key="pattern" value="yyyy-MM-dd" /-->
            </file>
            <db>
                    <add key="alias" value="qa_ms" />
            </db>
    </udl>
</configuration>
```

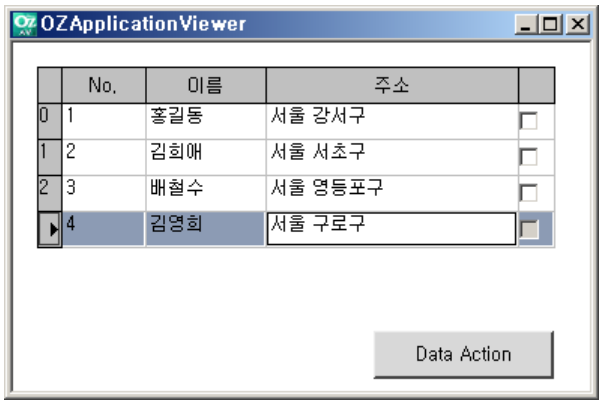
■ 데이터 액션을 수행하여 사용자 정의 로그 생성

- 닷넷 서버와 애플리케이션 디자이너를 구동한 후 애플리케이션 디자이너에서 구동된 서버에 접속합니다.

- 서버 리파지토리에서 데이터 액션을 수행하는 보고서를 열어 미리보기 합니다.



- 테이블에서 데이터를 추가한 후 [Data Action] 버튼을 클릭하여 데이터를 추가합니다.



- 서버가 설치된 경로의 /log 폴더 아래에 "UDL.log" 파일이 생성되었는지 확인합니다.

- 생성된 UDL.log 파일에 데이터 액션 시 실행된 쿼리문 및 ODI 정보 등이 로그로 기록되었는지 확인합니다.

```
[DEBUG] OZUserDefinedLogger.Trace - [DB Alias=odbc^^ozcar][IP
Address=127.0.0.1][User ID=admin][QueryType=Select][Query=select * from
REQ397]
[IsPrepared=False][QueryExeTime=156250][QueryState=True][ODIName=/DataAct
ion.odi][DataSetName=Sample_][DataStoreName=DB_]
[RequestParams=(ASP.NET_SessionId,45qruu55wds3w1451vibhs55)(ALL_HTTP,HTTP
...

[DEBUG] OZUserDefinedLogger.Trace - [DB Alias=odbc^^ozcar][IP
Address=127.0.0.1][User ID=admin][QueryType=Insert][Query=insert into
REQ397 (id,name,address) values (?,?,?)]
[IsPrepared=True][PreparedValues=3;3;3;][QueryExeTime=156250][QueryState=
True][ODIName=/DataAction.odi][DataSetName=Sample][DataStoreName=DB_]
[RequestParams=(ASP.NET_SessionId,45qruu55wds3w1451vibhs55)(ALL_HTTP,HTTP
...
```

UDL 예제 2 - 서버 로그를 데이터 베이스에 기록하는 방법

UDL을 이용하여 오즈 서버 로그를 데이터 베이스에 사용자가 정의한 타입의 형식으로 기록하는 방법에 대해 설명합니다.

■ **사용자 정의 로그 구현**

UDL을 사용하여 사용자 정의 로그를 데이터 베이스에 기록하는 클래스 파일을 만듭니다. 본 예제에서는 아래와 같은 java 소스 파일을 이용하여 oz.udl.db.OZUserDefinedLogger 클래스를 생성한 후 해당 클래스를 "ozudl.dll" 파일로 만들었습니다.

(해당 클래스 파일에서는 Web.config 파일에서 설정한 데이터 베이스의 "OZDBHISTORY" 테이블에 로그를 기록하도록 설정합니다.)

```
#region Using Statements
using System;
using System.Data;
using System.Text;
using System.Globalization;
using System.Configuration;
using System.Collections.Specialized;

using oz.udl;
#endregion

namespace oz.udl.db
{
```

```
  public class OZUserDefinedLogger : oz.udl.IUserDefinedLogger
   {
       private string _name = "DB UDL";

       private const string TableName = "OZDBHISTORY_NET";

       private static readonly string[] FieldNames =
       { "Time", "DbAlias", "IP", "UserID", "QueryType", "Query", "IsPrepared",
          "PreparedValues", "ExecTime", "QueryState", "OdiName", "DataSetName",
"DataStoreName",  "SessionInfo"};

       private static readonly int[] FieldSizes =
          { 0, 128, 27, 255, 6, int.MaxValue, 0, int.MaxValue, 0, 255, 255, 255,
255, int.MaxValue };

       private static readonly bool[] Nullable =
          {  false,  false,  true,  true,  true,  true,  true,  true,  false,  true,
false, false, false, true };

       // varchar's size must be specified
       private static readonly string[] FieldTypes =
          { "date", "varchar(128)", "varchar(27)", "varchar(255)", "varchar(6)",
"text", "bit", "text",
              "int",     "varchar(255)",     "varchar(255)",     "varchar(255)",
"varchar(255)", "text" };

       private static readonly DbType[] FieldDbTypes =
          {   DbType.DateTime,   DbType.String,   DbType.String,   DbType.String,
DbType.String,
              DbType.String,    DbType.Boolean,    DbType.String,    DbType.Int32,
DbType.String,
              DbType.String, DbType.String, DbType.String, DbType.String };

       private IDbCommand _cmd;
       private string _cmdText;
       public OZUserDefinedLogger() {}

       public void Init(oz.framework.db.OZConnection con, oz.framework.log.OZLog
log)
       {
          log.Debug(_name, "Called init");

          _cmd = con.Connection.CreateCommand();

          _cmd.CommandText = "select * from " + TableName;
          IDataReader reader = null;
          bool tableExists = true;
          try
          {
              reader = _cmd.ExecuteReader(CommandBehavior.SchemaOnly);
```

```
        }
        catch(Exception)
        {
            tableExists = false;
        }
        finally
        {
            if(null != reader)
                reader.Close();
        }
        StringBuilder sb = new StringBuilder();

        if(!!!tableExists)
        {
            sb.Append("CREATE TABLE ").Append(TableName);
            sb.Append('(');
            for(int i = 0; i < FieldNames.Length; i++)
            {
                sb.Append(FieldNames[i]).Append(' ');
                sb.Append(con.Vendor.GetFieldType(FieldTypes[i])).Append(' ');
                if(!!!Nullable[i]) sb.Append("NOT NULL");
                if(i != FieldNames.Length - 1) sb.Append(',');
            }
            sb.Append(')');

            _cmd.CommandText = sb.ToString();
            _cmd.ExecuteNonQuery();

            log.Debug(_name, "Success to create log table");
        }

        sb.Remove(0, sb.Length);
        sb.Append("INSERT INTO ").Append(TableName).Append(" VALUES(");
        for(int i = 0; i < FieldNames.Length; i++)
        {
            sb.Append(con.Vendor.MakeParameterName(i));

            if(i != FieldNames.Length - 1)
                sb.Append(',');
        }
        sb.Append(')');

        _cmdText = sb.ToString();
    }

    public string DbAlias
    {
        get
        {
            NameValueCollection                    values                    =
```

```
(NameValueCollection)ConfigurationSettings.GetConfig("udl/db");
            if(null == values)
               throw new Exception("Cannot find udl db configuration");
            return values["alias"];
         }
      }

      public          void          Trace(IUserDefinedLogTarget          target,
oz.framework.db.OZConnection con, oz.framework.log.OZLog log)
      {
         log.Debug(_name, "Called trace");
         IDbCommand cmd = con.Connection.CreateCommand();

         for(int i = 0; i < FieldNames.Length; i++)
         {
            IDbDataParameter param = cmd.CreateParameter();
            param.ParameterName = con.Vendor.MakeParameterName(i);
            param.DbType = FieldDbTypes[i];
            if(DbType.String == param.DbType)
               param.Size = FieldSizes[i];
            cmd.Parameters.Add(param);
         }

         cmd.CommandText = _cmdText;
         cmd.Prepare();

         StringBuilder sb = new StringBuilder();

         foreach(IDbDataParameter param in cmd.Parameters)
            param.Value = DBNull.Value;

         ((IDbDataParameter)cmd.Parameters[0]).Value = DateTime.Now;
         IRDBLogTarget rdbStore = target as IRDBLogTarget;
         if(null != rdbStore)
         {
            ((IDbDataParameter)cmd.Parameters[1]).Value = rdbStore.Alias;
            ((IDbDataParameter)cmd.Parameters[2]).Value = rdbStore.IPAddress;
            ((IDbDataParameter)cmd.Parameters[3]).Value = rdbStore.UserID;
            ((IDbDataParameter)cmd.Parameters[4]).Value = rdbStore.QueryType;
            ((IDbDataParameter)cmd.Parameters[5]).Value = rdbStore.Query;
            ((IDbDataParameter)cmd.Parameters[6]).Value = rdbStore.IsPrepared;
            if(rdbStore.IsPrepared)
            {
               sb.Remove(0, sb.Length);
               foreach(object o in rdbStore.PreparedQueryValues)
               {
                  sb.Append(null == o ? "null" : o is byte[] ? "binary" :
o.ToString());
                  sb.Append(";");
               }
```

```
                ((IDbDataParameter)cmd.Parameters[7]).Value = sb.ToString();
            }
            ((IDbDataParameter)cmd.Parameters[8]).Value                    =
rdbStore.QueryExecutionTime;
            ((IDbDataParameter)cmd.Parameters[9]).Value     =     null     ==
rdbStore.QueryState ? string.Empty : rdbStore.QueryState;
         }

         IDMLogTarget dataModule = target as IDMLogTarget;
         if(null != dataModule)
         {
            ((IDbDataParameter)cmd.Parameters[10]).Value = dataModule.ODIName;
            ((IDbDataParameter)cmd.Parameters[11]).Value                    =
dataModule.DataSetName;
            ((IDbDataParameter)cmd.Parameters[12]).Value                    =
dataModule.DataStoreName;
         }

         sb.Remove(0, sb.Length);
         NameValueCollection  parameters  =  null  !=  target.HttpRequest  ?
target.HttpRequest.Params : null;
         if(null != parameters)
         {
            sb.Append("[RequestParams=");

            foreach(string key in parameters.AllKeys)
            {
               sb.AppendFormat("({0},{1})", key, parameters[key]);
            }
            sb.Append("]");
         }

         System.Web.SessionState.HttpSessionState    session    =    null    !=
target.HttpContext ? target.HttpContext.Session : null;
         if(null != session)
         {
            sb.Append("[SessionAttrs=");
            foreach(string key in session.Keys)
            {
               sb.AppendFormat("({0},{1})", key, session[key]);
            }
         }
         ((IDbDataParameter)cmd.Parameters[13]).Value = sb.ToString();
         log.Debug(sb.ToString());

         cmd.ExecuteNonQuery();
         log.Debug("Success to insert log information");
      }
   }
```

```
   public class OZUserDefinedLogger2
   {
      private OZUserDefinedLogger _logger;

      public OZUserDefinedLogger2()
      {
         _logger = new OZUserDefinedLogger();
      }

      [OZUDLInitialize]
      public void Init(oz.framework.db.OZConnection con, oz.framework.log.OZLog
log)
      {
         _logger.Init(con, log);
      }

      [OZUDLTrace]
      public          void          Trace(IUserDefinedLogTarget          target,
oz.framework.db.OZConnection con, oz.framework.log.OZLog log)
      {
         _logger.Trace(target, con, log);
      }
   }
}
```

■ **서버 환경 설정**

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

```
#---------------------------------
# configuraion of OZ User Defined log
#---------------------------------

OZ_USER_DEFINED_LOG.Active=true
OZ_USER_DEFINED_LOG.Class=oz.udl.db.OZUserDefinedLogger
```

- 닷넷 서버가 설치된 경로의 Web.config 파일을 텍스트 편집기로 열어 사용자 정의 로그를 기록할 데이터 베이스와 로그 형식을 아래와 같이 설정한 후 저장합니다.
  (본 예제에서는 오즈 서버의 db.properties 파일에 정의된 DB 커넥션 풀 중 "Sample " 데이터 베이스의 로그를 기록하도록 설정합니다.).

```
<?xml version="1.0" encoding="utf-8" ?>
    <configuration>
    <configSections>
<!--.Net Framework 2.0 버전 이하를 사용할 경우 반드시 아래와 같이 .Net Framework
버전을 명시하여야합니다.-->
            <sectionGroup name="udl">
```

```
                    <section name="file"
type="System.Configuration.NameValueSectionHandler, System,
Version=1.0.5000.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"/>
                    <section name="db"
type="System.Configuration.NameValueSectionHandler, System,
Version=1.0.5000.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"/>
            </sectionGroup>
    </configSections>
  <system.web>
<httpRuntime maxRequestLength="100000"  />
 </system.web>
    <udl>
            <file>
                    <add key="path" value="C:/Program Files/Forcs/50/OZ
Server.NET 5.0/logs/UDL.Log" />
                    <add key="layout" value="%d{yyyy-MM-dd HH:mm:ss.fff}
[%-5p] %C{1}.%M - %m%n" />
                    <add key="daily" value="false" />
                    <add key="size" value="100KB" />
                    <add key="maxBackup" value="5" />
                    <!--add key="daily" value="true" />
                    <add key="pattern" value="yyyy-MM-dd" /-->
            </file>
            <db>
                    <add key="alias" value="Sample" />
            </db>
    </udl>
</configuration>
```

- OZ_HOME/conf/db.properties 파일에 oz.udl.db.OZUserDefinedLogger에서 사용할 데이터 베이스 커넥션 풀을 추가합니다.

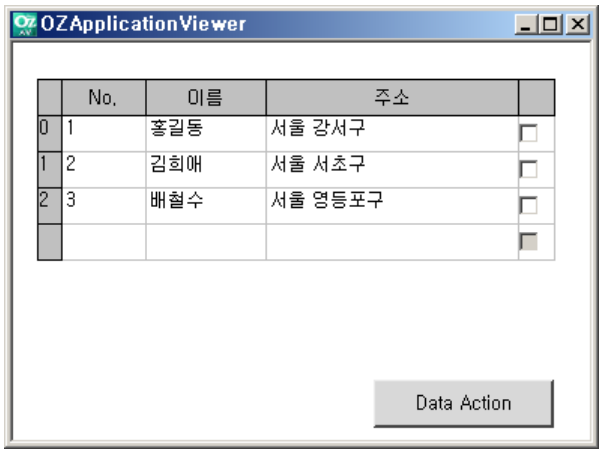
```
#
#  Sample
#
Sample.vendor=MSSQL
Sample.serverAddress=127.0.0.1
Sample.user=user
Sample.password=userpw
Sample.portNo=1433
Sample.dbName=DBName
Sample.maxconns=20
Sample.initconns=5
Sample.timeout=5
Sample.doConnectionCheck=false
```

■ **데이터 액션을 수행하여 사용자 정의 로그 생성**

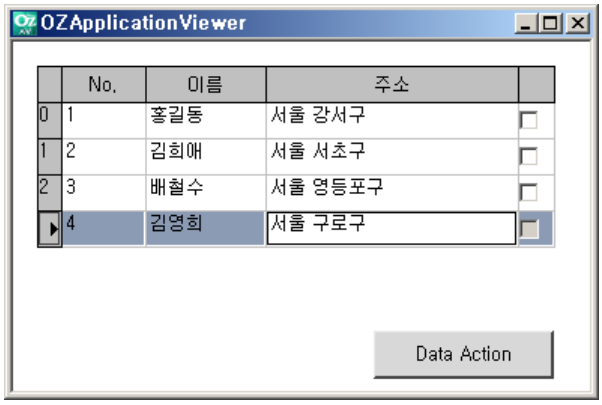
- 닷넷 서버와 애플리케이션 디자이너를 구동한 후 애플리케이션 디자이너에서 구동된

서버에 접속합니다.

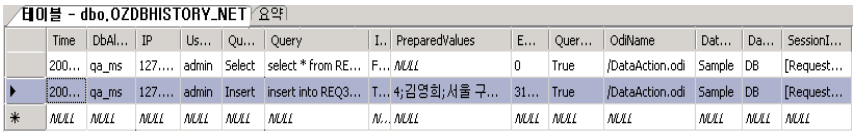
- 서버 리파지토리에서 데이터 액션을 수행하는 보고서를 열어 미리보기 합니다.



- 테이블에서 데이터를 추가한 후 [Data Action] 버튼을 클릭하여 데이터를 추가합니다.



- 설정한 데이터 베이스 테이블에 데이터 액션 시 실행된 쿼리문 및 ODI 정보 등이 로그로 기록되었는지 확인합니다.



UDL 예제 3 - 모니터 로그를 파일과 데이터 베이스에 기록하는 방법

닷넷 서버에서 OZ UDL을 사용하여 monitor.log 파일과 데이터 베이스에 사용자 정의 모니터 로그를 남기는 방법에 대해 설명합니다.

■ **사용자 정의 로그 구현**

UDL을 사용하여 사용자 정의 모니터 로그를 monitor.log 파일과 데이터 베이스에 기록하는 클래스 파일을 만듭니다. 본 예제에서는 아래와 같은 java 소스 파일을 이용하여 oz.udl.db.OZUserDefinedMonitorLogger 클래스를 생성한 후 해당 클래스를 "ozudl.dll" 파일로 만들었습니다.

(해당 클래스 파일에서는 오즈 서버의 db.properties 파일에 정의된 DB 커넥션 풀 중 "Sample" 데이터 베이스의 "OZ_MONITOR_JAVA" 테이블에 로그를 기록하도록 설정합니다.)

```
#region Using Statements
using System;
using System.Data;
using System.Text;
using System.Globalization;
using System.Configuration;
using System.Collections.Specialized;

using oz.udl;
#endregion

namespace oz.udl.db
{
public class OZUserDefinedMonitorLogger : oz.udl.IUserDefinedLogger
   {
       private string _name = "DB UDL";
       private static object s_lock = new object();

       private const string TableName = "OZ_MONITOR_NET";

       private static readonly string[] FieldNames =
       {
              "MARK", "THREAD_ID", "SERVICE_TIME", "FREE_MEMORY", "TOTAL_MEMORY",
"SERVICE_CODE", "SERVICE_STATUS",
          "SERVICE_PARAMETERS", "DB_SESSION_ID", "EXECUTION_TIME", "CLIENT_IP",
"SESSION_INFO"};

       private static readonly int[] FieldSizes =
          { 5, 0, 0, 0, 0, 0, 0, int.MaxValue, 255, 0, 20, int.MaxValue };

       private static readonly bool[] Nullable =
          { false, false, false, true, true, true, true, true, true, true, true,
true };

       // varchar's size must be specified
       private static readonly string[] FieldTypes =
          {
                 "varchar(5)", "int", "bigint", "bigint", "bigint", "smallint",
```

```
"smallint", "text", "varchar(255)", "bigint", "varchar(50)", "text" };

      private static readonly DbType[] FieldDbTypes =
         {
            DbType.String,     DbType.Int32,     DbType.Int64,     DbType.Int64,
DbType.Int64, DbType.Int16, DbType.Int16,
            DbType.String,    DbType.String,    DbType.Int64,    DbType.String,
DbType.String };

      private IDbCommand _cmd;

      private string _cmdText;

      public OZUserDefinedMonitorLogger()
      {
      }

      public void Init(oz.framework.db.OZConnection con, oz.framework.log.OZLog
log)
      {
         log.Debug(_name, "Called init");

         _cmd = con.Connection.CreateCommand();

         _cmd.CommandText = "select * from " + TableName;
         IDataReader reader = null;
         bool tableExists = true;
         try
         {
            reader = _cmd.ExecuteReader(CommandBehavior.SchemaOnly);
         }
         catch(Exception)
         {
            tableExists = false;
         }
         finally
         {
            if(null != reader)
               reader.Close();
         }

         StringBuilder sb = new StringBuilder();
         //사용자 정의 모니터 로그를 기록할 테이블과 테이블에 데이터 추가 시 실행될 쿼리문
설정
         if(!!!tableExists)
         {
            sb.Append("CREATE TABLE ").Append(TableName);
            sb.Append('(');
            for(int i = 0; i < FieldNames.Length; i++)
            {
```

```
                sb.Append(FieldNames[i]).Append(' ');
                sb.Append(con.Vendor.GetFieldType(FieldTypes[i])).Append(' ');
                if(!!!Nullable[i]) sb.Append("NOT NULL");
                if(i != FieldNames.Length - 1) sb.Append(',');
            }
            sb.Append(')');

            _cmd.CommandText = sb.ToString();
            _cmd.ExecuteNonQuery();

            log.Debug(_name, "Success to create log table");
         }

         sb.Remove(0, sb.Length);
         sb.Append("INSERT INTO ").Append(TableName).Append(" VALUES(");
         for(int i = 0; i < FieldNames.Length; i++)
         {
            sb.Append(con.Vendor.MakeParameterName(i));

            if(i != FieldNames.Length - 1)
               sb.Append(',');
         }
         sb.Append(')');

         _cmdText = sb.ToString();
         log.Debug("Query : " + _cmdText);
      }
      //UDL을 DB로 할 경우 OZ Server에 Pool된 alias이름
      public string DbAlias{ get{ return "Sample"; } }

      public          void          Trace(IUserDefinedLogTarget          target,
oz.framework.db.OZConnection con, oz.framework.log.OZLog log)
      {
         log.Debug(_name, "Called trace");
         IDbCommand cmd = con.Connection.CreateCommand();

         lock(s_lock)
         {

            for(int i = 0; i < FieldNames.Length; i++)
            {
               IDbDataParameter param = cmd.CreateParameter();
               param.ParameterName = con.Vendor.MakeParameterName(i);
               param.DbType = FieldDbTypes[i];
               if(DbType.String == param.DbType)
                  param.Size = FieldSizes[i];
               cmd.Parameters.Add(param);
            }

            cmd.CommandText = _cmdText;
```

```
            cmd.Prepare();
         }

         foreach(IDbDataParameter param in cmd.Parameters)
            param.Value = DBNull.Value;
         //로그에 기록할 타겟 구현
         IMonitorLogTarget monitor = target as IMonitorLogTarget;
         if(null != monitor)
         {
            ((IDbDataParameter)cmd.Parameters[0]).Value = monitor.Mark;
            ((IDbDataParameter)cmd.Parameters[1]).Value = monitor.ThreadID;
            ((IDbDataParameter)cmd.Parameters[2]).Value = monitor.ServiceTime;
            ((IDbDataParameter)cmd.Parameters[3]).Value = monitor.FreeMemory;
            ((IDbDataParameter)cmd.Parameters[4]).Value = monitor.TotalMemory;
            ((IDbDataParameter)cmd.Parameters[5]).Value = monitor.ServiceCode;
            ((IDbDataParameter)cmd.Parameters[6]).Value                        =
monitor.ServiceStatus;
            ((IDbDataParameter)cmd.Parameters[7]).Value                        =
monitor.ServiceParameter;
            ((IDbDataParameter)cmd.Parameters[8]).Value = monitor.DBSessionID;
            ((IDbDataParameter)cmd.Parameters[9]).Value = monitor.ExecuteTime;
            ((IDbDataParameter)cmd.Parameters[10]).Value = monitor.IPAddress;
         }

         StringBuilder sb = new StringBuilder();

         NameValueCollection  parameters  =  null  !=  target.HttpRequest  ?
target.HttpRequest.Params : null;
         if(null != parameters)
         {
            sb.Append("[RequestParams=");

            foreach(string key in parameters.AllKeys)
            {
               sb.AppendFormat("({0},{1})", key, parameters[key]);
            }
            sb.Append("]");
         }

         System.Web.SessionState.HttpSessionState   session   =   null   !=
target.HttpContext ? target.HttpContext.Session : null;
         if(null != session)
         {
            sb.Append("[SessionAttrs=");
            foreach(string key in session.Keys)
            {
               sb.AppendFormat("({0},{1})", key, session[key]);
            }
         }
         ((IDbDataParameter)cmd.Parameters[11]).Value = sb.ToString();
```

```
            cmd.ExecuteNonQuery();
            log.Debug("Success to insert log information");
        }
    }

    public class OZUserDefinedMonitorLogger2
    {
        private OZUserDefinedMonitorLogger _logger;

        public OZUserDefinedMonitorLogger2()
        {
            _logger = new OZUserDefinedMonitorLogger();
        }

        [OZUDLInitialize]
        public void Init(oz.framework.db.OZConnection con, oz.framework.log.OZLog
log)
        {
            _logger.Init(con, log);
        }

        [OZUDLTrace]
        public          void          Trace(IUserDefinedLogTarget          target,
oz.framework.db.OZConnection con, oz.framework.log.OZLog log)
        {
            _logger.Trace(target, con, log);
        }
    }
}
```

■ **서버 환경 설정**

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

```
#---------------------------------
# configuraion of OZ User Defined Monitor log
#---------------------------------

OZ_UDL_MONITOR.Active=true
OZ_UDL_MONITOR.Class=oz.udl.db.OZUserDefinedMonitorLogger
```

- db.properties 파일에 oz.udl.db.OZUserDefinedLoggerForDB에서 사용할 데이터베이스 커넥션 풀을 추가합니다.

```
#
#  Sample
#
```

```
Sample.vendor=MSSQL
Sample.serverAddress=127.0.0.1
Sample.user=user
Sample.password=userpw
Sample.portNo=1433
Sample.dbName=DBName
Sample.maxconns=20
Sample.initconns=5
Sample.timeout=5
Sample.doConnectionCheck=false
```

■ **사용자 정의 모니터 로그 생성**

- 닷넷 서버와 애플리케이션 디자이너를 구동한 후 애플리케이션 디자이너에서 구동된 서버에 접속합니다.

- 서버 리파지토리에서 보고서를 열기하여 모니터 로그에 요청 사항을 기록합니다 합니다.

- 서버가 설치된 경로의 /log/monitor.log 파일에 보고서 요청 시 클라이언트 IP 및 서비스 상태 등이 로그로 기록되었는지 확인합니다.

```
start 16 11840488774675572048977921098953785344 null null null null nullnull

end 16 11840488774675572048977921098953785344 174 9001 item
name;dac_prepared.oza,item type;OZA,category name;/Test 127.0.0.1 0

start 16 11840488774985571294003201098953785344 null null null null null
null

end 16 11840488775145571168174081098953785344 196 9001 item
name;dac_prepared.oza,item type;OZA,category name;/Test,compressed;True
127.0.0.1 16
```

- 설정한 데이터 베이스 테이블에 보고서 요청 시 클라이언트 IP 및 서비스 상태 등이 로그로 기록되었는지 확인합니다.

	MARK	THRE...	SERVICE_TIME	FREE_MEM...	TOTAL_MEM...	SE...	SER...	SERVICE_PARAMET...	DB...	E...	CLIEN...	SESSION_INFO
	start	16	1184048968483	544055754752	1098953785344	-1	-1			-1	127.0.0.1	[RequestParams...
	end	16	1184048968483	544055754752	1098953785344	174	9001	item name;Car.oza,i...		0	127.0.0.1	[RequestParams...
	start	16	1184048877342	557628522496	1098953785344	-1	-1			-1	127.0.0.1	[RequestParams...
	end	16	1184048968514	543887982592	1098953785344	174	9001	item name;Car.oza,i...		0	127.0.0.1	[RequestParams...

Ⅴ. 오즈 리파지토리 구현

- Interface Repository
- Class RepositoryEx
- 오즈 리피지토리 구현

본 장에서는 오즈 리파지토리를 구현하는 방법에 대해 설명합니다.

Interface Repository

Interface Summary

// NameSpace : oz.framework.repositoryex

■ 기본 리파지토리 인터페이스

- public interface IRepository

■ 아이템 관련 리파지토리 인터페이스

- public interface IRepositoryCategory
- public interface IRepositoryItem
- public interface IRepositoryHistory

■ 사용자 관련 리파지토리 인터페이스

- public interface IRepositoryGroup
- public interface IRepositoryUser
- public interface IRepositoryMultiLoginUser

■ Exception 관련 클래스

- public class OZRepositoryException extends Exception

// 패키지명 : oz.framework.repositoryex.auth

■ 권한 관련 인터페이스

- public interface IRepositoryAuthGroupCategory
- public interface IRepositoryAuthGroupItem
- public interface IRepositoryAuthUserCategory
- public interface IRepositoryAuthUserItem

// 패키지명 : oz.framework.repositoryex.info

■ 리파지토리 정보(bean) 관련 인터페이스

- public interface ICategoryInfo

- public interface IItemInfo
- public interface IHistoryInfo
- public interface IGroupInfo
- public interface IUserInfo
- public interface ILoginInfo

Interface Method Summary

// IRepository

- IStringDictionary Property{set;}
- int Version{get;}
- void Open ()
- void Close()
- object Login(string userName, string password, ILoginInfo loginInfo, HttpContext ctx)
- void UpdateSesssionState(object session, SessionState state)
- bool Logout(object session, string username, ILoginInfo loginInfo, HttpContext ctx)
- IRepositoryCategory Category{get;}
- IRepositoryItem Item{get;}
- IRepositoryHistory History{get;}
- IRepositoryGroup Group{get;}
- IRepositoryUser User{get;}
- IRepositoryMultiLoginUser MultiLoginUser{get;}
- IRepositoryGroupAuthorityToCategory GroupAuthorityToCategory{get;}
- IRepositoryGroupAuthorityToItem GroupAuthorityToItem{get;}
- IRepositoryUserAuthorityToCategory UserAuthorityToCategory{get;}
- IRepositoryUserAuthorityToItem UserAuthorityToItem{get;}

// IRepositoryCategory

- IRepository Repository{get; set;}
- CategoryAccess AccessType{get;}
- string[] CreateItems(object session, string[] itemNames, string[] descriptions, string[] categoryIDs, Stream[] items, string comment, OZErrorCode[] errCodes, string[] errMsgs)

- string[] CreateCategories(object session, string[] categoryNames, string[] parentCategoryIDs, string comment, OZErrorCode[] errCodes, string[] errMsgs)
- string ModifyCategoryName(object session, string categoryID, string newCategoryName, string comment)
- bool[] DeleteCategories(object session, string[] categoryIDs, bool[] toBeDestroyed, string comment, OZErrorCode[] errCodes, string[] errMsgs)
- bool[] UndeleteCategories(object session, string[] categoryIDs, string comment, OZErrorCode[] errCodes, string[] errMsgs)
- int GetItemCount(object session, string categoryID)
- IItemInfo[] GetItemInfos(object session, string categoryID)
- string GetCategoryID(object session, string itemID)
- ICategoryInfo GetCategoryInfo(object session, string categoryID)
- IItemInfo[] GetDeletedItemInfos(object session, string categoryID)
- IItemInfo[] SearchItem(object session, string categoryID, SearchOption option, string criteria)
- ICategoryInfo[] GetCategoryInfos(object session, string categoryID)
- bool TransferItems(object session, string[] itemIDs, string categoryID)
- bool TransferCategory(object session, string categoryID, string targetCategoryID)

// IRepositoryItem

- IRepository Repository{get; set;}
- ItemAccess AccessType{get;}
- string[] CreateItems(object session, string[] itemNames, string[] descriptions, Stream[] items, string comment, OZErrorCode[] errCodes, string[] errMsgs)
- string ModifyItemName(object session, string itemID, string newItemName, string comment)
- bool[] DeleteItems(object session, string[] itemIDs, bool[] toBeDestroyed, string comment, OZErrorCode[] errCodes, string[] errMsgs)
- bool[] UndeleteItems(object session, string[] itemIDs, string comment, OZErrorCode[] errCodes, string[] errMsgs)
- bool ModifyItemDescription(object session, string itemID, string description)
- IItemInfo GetItemInfo(object session, string itemID)

- bool HasItem(object session, string itemID)
- Stream[] GetItems(object session, string[] itemIDs, OZErrorCode[] errCodes, string[] errMsgs)
- Stream[] GetItems(object session, string[] itemIDs, long[] modifiedTimes, ItemOptions[] options, OZErrorCode[] errCodes, string[] errMsgs)
- Stream[] CheckOut(object session, string[] itemIDs, string[] localCheckOutFolders, long[] localFileTimes, ItemOptions[] options, OZErrorCode[] errCodes, string[] errMsgs)
- bool[] CheckIn(object session, string[] itemIDs, Stream[] items, string comment, bool[] keepCheckOut, OZErrorCode[] errCodes, string[] errMsgs)
- Stream[] UndoCheckOut(object session, string[] itemIDs, ItemOptions[] options, OZErrorCode[] errCodes, string[] errMsgs)
- bool[] IsCheckOutUser(object session, string[] itemIDs)

// IRepositoryHistory

- IRepository Repository{get; set;}
- HistoryAccess AccessType{get;}
- bool RollBack(object session, string itemID, int version, string comment)
- Stream GetItem(object session, string itemID, int version)
- IHistoryInfo[] GetHistoryInfos(object session, string itemID)
- IHistoryInfo[] GetDeletedItemHistoryInfos(object session, string itemID)
- bool RemoveHistory(object session, string itemID, int version)

// IRepositoryGroup

- IRepository Repository{get; set;}
- GroupAccess AccessType{get;}
- string CreateGroup(object session, string groupName, string parentGroupID, string description)
- string ModifyGroupName(object session, string groupID, string groupName)
- bool ModifyGroupDescription(object session, string groupID, string description)
- bool DeleteGroup(object session, string groupID)
- string CreateUser(object session, string userName, string password, string groupID, string description)
- IUserInfo[] GetUserInfos(object session, string groupID)

- IGroupInfo GetGroupInfo(object session, string groupID)
- IGroupInfo[] GetSubGroupInfos(object session, string groupID)
- IGroupInfo GetParentGroupInfo(object session, string groupID)
- string GetGroupID(object session, string userID)
- bool AddGroupAdministrator(object session, string userID, string groupID)
- bool RemoveGroupAdministrator(object session, string userID, string groupID)
- bool IsGroupAdministrator(object session, string userID, string groupID)
- IUserInfo[] GetAdministrators(object session, string groupID)
- bool TransferUser(object session, string userID, string newGroupID)
- bool TransferGroup(object session, string groupID, string targetGroupID)
- bool TransferGroup(object session, string groupID, string targetGroupID)

// IRepositoryUser

- IRepository Repository{get; set;}
- UserAccess AccessType{get;}
- string CreateUser(object session, string userName, string password, string description)
- string ModifyName(object session, string userID, string newUserName)
- bool ModifyPassword(object session, string userID, string oldPassword, string newPassword)
- bool ModifyDescription(object session, string userID, string description)
- bool DeleteUser(object session, string userID)
- IUserInfo GetUserInfo(object session, string userID)
- bool CheckPassword(object session, string userID, string password)
- IUserInfo[] GetUserInfos(object session)
- bool IsAdministrator(object session, string userID)

// IRepositoryMultiLoginUser

- IRepository Repository{get; set;}
- MultiLoginAccess AccessType{get;}
- void DisableLogin(object session, string userID)
- void EnableLogin(object session, string userID)
- bool IsLoginEnabled(object session, string userID)

// IRepositoryAuthGroupCategory

- IRepository Repository{get; set;}
- AuthorityAccess AccessType{get;}
- bool Modify(object session, string categoryID, string groupID, Authority permission)
- Authority GetAuthority(object session, string groupID, string categoryID)
- ICategoryInfo[] GetCategoryInfos(object session, string groupID, string categoryID, Authority permission)
- IGroupInfo[] GetGroupInfos(object session, string categoryID, Authority permission)
- IItemInfo[] GetItemInfos(object session, string groupID, string categoryID, Authority permission)

// IRepositoryAuthGroupItem

- IRepository Repository{get; set;}
- AuthorityAccess AccessType{get;}
- bool Modify(object session, string groupID, string itemID, Authority permission)
- Authority GetAuthority(object session, string groupID, string itemID)
- IGroupInfo[] GetGroupInfos(object session, string itemID, Authority permission)
- IItemInfo[] GetItemInfos(object session, string groupID, Authority permission)

// IRepositoryAuthUserCategory

- IRepository Repository{get; set;}
- AuthorityAccess AccessType{get;}
- bool Modify(object session, string userID, string categoryID, Authority permission)
- Authority GetAuthority(object session, string userID, string categoryID)
- ICategoryInfo[] GetCategoryInfos(object session, string userID, string categoryID, Authority permission)
- IUserInfo[] GetUserInfos(object session, string categoryID, Authority permission)
- IItemInfo[] GetItemInfos(object session, string userID, string categoryID, Authority permission)

// IRepositoryAuthUserItem

- **IRepository Repository{get; set;}**
- **AuthorityAccess AccessType{get;}**
- **bool Modify(object session, string userID, string itemID, Authority permission)**
- **Authority GetAuthorigy(object session, string userID, string itemID)**
- **IUserInfo[] GetUserInfos(object session, string itemID, Authority permission)**
- **IItemInfo[] GetItemInfos(object session, string userID, Authority permission)**

Interface Method Detail

/ / IRepository Interface Method Summary

- **Property**

Prototype	`oz.util.IStringDictionary Property{set;}`
Definition	리파지토리 설정을 변경합니다. "repository.properties"에 설정되어 있는 값을 변경할 수 있으며, 설정할 수 있는 Key와 값은 아래 표를 참조하시기 바랍니다.
Argument	*IStringDictionary* 리파지토리 설정 속성

- **Version**

Prototype	`int Version{get;}`
Definition	리파지토리 버전을 정의한 후 정의한 버전을 가져옵니다.

- **Open**

Prototype	`void Open()`
Definition	서버 구동 시 리파지토리를 오픈합니다.

- **Close**

Prototype	`void Close()`
Definition	서버 종료 시 리파지토리를 종료합니다.

- **Login**

Prototype	`object Login(string username, string password, ILoginInfo loginInfo, HttpContext ctx)`
Definition	클라이언트를 호출할 때 해당 메소드를 호출하여 로그인한 세션 정보를 가져옵

	니다.	
Argument	*UserName*	접속한 사용자 이름
	password	접속한 사용자의 패스워드
	loginInfo	로그인 정보 ※ 참고사항 : 자세한 로그인 정보는 ILoginInfo 인터페이스를 참조하시기 바랍니다.
	ctx	HTTP 접속 시 현재 컨텍스트 ※ 참고사항 : 윈도우즈 서버스 형태의 서버인 경우 null 값을 설정합니다..

■ UpdateSesssionState

Prototype	`void UpdateSesssionState(object session, SessionState state)`	
Definition	클라이언트를 호출할 때 해당 메소드를 호출하여 로그인한 세션 정보를 가져옵니다.	
Argument	*session*	세션 ID
	state	서버 상태 • None • BareServer • FromServer

■ Logout

Prototype	`bool Logout(object session, string userName, ILoginInfo loginInfo, HttpContext ctx)`	
Definition	지정한 사용자 ID에 해당하는 사용자를 로그 아웃 즉, 접속 해제시키며 로그아웃 성공 여부를 가져옵니다.	
Argument	*session*	세션 ID
	userName	사용자 이름
	loginInfo	로그인 정보 (자세한 정보는 해당 인터페이스 참조)
	ctx	HTTP 접속 시 현재 컨텍스트 ※ 참고사항 : 윈도우즈 서버스 형태의 서버인 경우 null 값을 설정합니다..

■ Category

Prototype	`IRepositoryCategory Category{get;}`
Definition	카테고리 인터페이스를 구현한 경우 구현된 객체를 가져옵니다.

■ Item

Prototype	`IRepositoryItem Item{get;}`

Definition	아이템 인터페이스를 구현한 경우 해당 구현된 객체를 가져옵니다.

■ **ItemHistory**

Prototype	IRepositoryHistory ItemHistory{get;}
Definition	아이템 히스토리 인터페이스를 구현한 경우 해당 구현된 객체를 가져옵니다.

■ **Group**

Prototype	IRepositoryGroup Group{get;}
Definition	그룹 인터페이스를 구현한 경우 해당 구현된 객체를 가져옵니다.

■ **User**

Prototype	IRepositoryUser User{get;}
Definition	유저 인터페이스를 구현한 경우 해당 구현된 객체를 가져옵니다.

■ **MultiLoginUser**

Prototype	IRepositoryMultiLoginUser MultiLoginUser{get;}
Definition	멀티 로그인 유저 인터페이스를 구현한 경우 해당 구현된 객체를 가져옵니다.

■ **GroupAuthorityToCategory**

Prototype	IRepositoryGroupAuthorityToCategory GroupAuthorityToCategory{get;}
Definition	카테고리에 대한 그룹의 권한 인터페이스를 구현한 경우 해당 구현된 클래스를 가져옵니다.

■ **GruopAuthorityToItem**

Prototype	IRepositoryGroupAuthorityToItem GruopAuthorityToItem{get;}
Definition	아이템에 대한 그룹의 권한 인터페이스를 구현한 경우 해당 구현된 객체를 가져옵니다.

■ **UserAuthorityToCtegory y**

Prototype	IRepositoryUserAuthorityToCategory UserAuthorityToCtegory{get;}
Definition	카테고리에 대한 사용자의 권한 인터페이스를 구현한 경우 해당 구현된 클래스를 가져옵니다.

■ **UserAuthorityToItem**

Prototype	IRepositoryUserAuthorityToItem UserAuthorityToItem

Definition	아이템에 대한 사용자의 권한 인터페이스를 구현한 경우 해당 구현된 클래스를 가져옵니다.

// IRepositoryCategory Interface Method Summary

■ **AccessType**

Prototype	`CategoryAccess AccessType{get;}`
Definition	카테고리 인터페이스에 접근 가능한 메소드를 숫자값으로 가져옵니다. None = 0x00000000 All = 0x7FFFFFFF CreateCategories = 0x00000002 ModifyCategoryName = 0x00000004 DeleteCategories = 0x00000008 UndeleteCategories = 0x00000010 HasItem = 0x00000020 GetItemCount = 0x00000040 GetItemInfos = 0x00000080 GetCategoryID = 0x00000100 GetCategoryInfo = 0x00000200 GetDeletedItemInfos = 0x00000400 SearchItem = 0x00000800, GetCategoryInfos = 0x00001000 TransferItems = 0x00002000 TransferCategory = 0x00004000

■ **Repository**

Prototype	`IRepository Repository{get; set;}`
Definition	IRepostiroy 인터페이스를 구현한 객체를 가져옵니다.

■ **CreateItems**

Prototype	`string[] CreateItems(object session, string[] itemNames, string[] descriptions, string[] categoryIDs, Stream[] items, string comment, OZErrorCode[] errCodes, string[] errMsgs)`	
Definition	새로운 아이템을 생성하고 생성된 아이템의 ID를 반환합니다.	
Argument	*session*	세션 ID
	itemNames	새로 생성할 아이템의 ID
	descriptions	새로 생성할 아이템의 설명 내용
	categoryIDs	새로 생성할 아이템이 카테고리 ID
	items	새로 생성할 아이템의 입력 스트림
	comment	주석문
	errCodes	에러 코드

	errMsgs	에러 메시지

■ **CreateCategories**

Prototype	`string[] CreateCategories(object session, string[] categoryNames, string[] parentCategoryIDs, string comment, OZErrorCode[] errCodes, string[] errMsgs)`	
Definition	카테고리를 새로 생성하고 생성된 카테고리의 ID를 반환합니다.	
Argument	*session*	세션 ID
	categoryNames	생성할 카테고리의 이름
	parentCategoryIDs	생성할 카테고리의 상위 카테고리 ID
	comment	주석문
	errCodes	에러 코드
	errMsgs	에러 메시지

■ **ModifyCategoryName**

Prototype	`string ModifyCategoryName(object session, string categoryID, string newCategoryName, string comment)`	
Definition	지정한 카테고리 ID에 해당하는 카테고리의 이름을 변경합니다.	
Argument	*session*	세션 ID
	categoryID	이름을 변경할 카테고리의 ID
	newCategoryName	변경할 카테고리 이름
	comment	주석문

■ **DeleteCategories**

Prototype	`bool[] DeleteCategories(object session, string[] categoryIDs, bool[] toBeDestroyed, string comment, OZErrorCode[] errCodes, string[] errMsgs)`	
Definition	지정한 카테고리 ID에 해당하는 카테고리를 삭제합니다.	
Argument	*session*	세션 ID
	categoryIDs	삭제할 카테고리의 ID
	toBeDestroyed	카테고리를 영구 삭제할 지 여부
	comment	주석문
	errorCodes	에러 코드
	errorMsgs	에러 메시지

■ **UndeleteCategories**

Prototype	bool[] UndeleteCategories(object session, string[] categoryIDs, string comment, OZErrorCode[] errCodes, string[] errMsgs)	
Definition	삭제된 카테고리를 복원하고 복원 성공 여부를 가져옵니다. ※ 주의사항 : 서버 버전이 2007년 09월 01일 이전 버전일 경우 해당 메소드가 지원되지 않습니다. ※ 참고사항 : 카테고리 삭제 시 "isDestroys=false"로 설정하였을 경우 삭제된 카테고리의 복원이 가능합니다.	
Argument	*session*	세션 ID
	CategoryIDs	복원할 카테고리의 ID
	comment	주석문
	errorCode	에러 코드
	errorMsg	에러 메시지

■ **GetItemCount**

Prototype	int GetItemCount(object session, string categoryID)	
Definition	지정한 카테고리에 속해 있는 모든 아이템의 개수를 가져옵니다.	
Argument	*session*	세션 ID
	categoryID	아이템의 개수를 얻을 카테고리의 ID

■ **GetItemInfos**

Prototype	IItemInfo[] GetItemInfos(object session, string categoryID)	
Definition	지정한 카테고리에 속해 있는 모든 아이템 정보를 가져옵니다.	
Argument	*session*	세션 ID
	categoryID	아이템의 정보를 얻을 카테고리의 ID

■ **GetCategoryInfos**

Prototype	ICategoryInfo[] GetCategoryInfos(object session, string categoryID)	
Definition	지정한 카테고리의 정보를 가져옵니다.	
Argument	*session*	세션 ID
	categoryID	정보를 가져올 카테고리 ID

■ **GetCategoryID**

Prototype	string GetCategoryID(object session, string itemID)	
Definition	지정한 아이템이 존재하는 카테고리의 ID를 반환합니다.	
Argument	*session*	세션 ID

	itemID	카테고리의 ID를 얻을 아이템 ID

■ **GetCategoryInfo**

Prototype	`ICategoryInfo GetCategoryInfo(object session, string categoryID)`	
Definition	지정한 카테고리의 정보를 가져옵니다.	
Argument	*session*	세션 ID
	categoryID	정보를 가져올 카테고리 ID

■ **GetDeletedItemInfos**

Prototype	`IItemInfo[] GetDeletedItemInfos(object session, string categoryID)`	
Definition	지정한 카테고리의 삭제된 아이템 정보를 가져옵니다. ※ 참고사항 : "toBeDestroyed =false"로 설정하여 삭제한 아이템의 정보만 가져옵니다.	
Argument	*session*	세션 ID
	categoryID	삭제된 아이템 정보를 가져올 카테고리 ID

■ **TransferItems**

Prototype	`bool TransferItems(object session, string[] itemIDs, string targetCategoryID)`	
Definition	지정한 아이템의 카테고리를 이동하고 카테고리 이동 성공 여부를 가져옵니다.	
Argument	*session*	세션 ID
	itemIDs	카테고리를 이동할 아이템 ID
	targetCategoryID	이동할 카테고리 ID

■ **TransferCategory**

Prototype	`bool TransferCategory(object session, string categoryID, string targetCategoryID)`	
Definition	지정한 카테고리를 다른 카테고리로 이동하고 카테고리 이동 성공 여부를 가져옵니다.	
Argument	*session*	세션 ID
	categoryID	카테고리를 이동할 카테고리 ID
	targetCategoryID	이동할 카테고리 ID

/ / IOZRepositoryItem

AccessType

Prototype	`CategoryAccess AccessType{get;}`
Definition	아이템 인터페이스에 접근 가능한 메소드를 숫자값으로 가져옵니다. None = 0x00000000 All = 0x7FFFFFFF CreateItems = 0x00000001 ModifyItemName = 0x00000002 DeleteItems = 0x00000004 UndeleteItems = 0x00000008 ModifyItemDescription = 0x00000010 GetItemInfo = 0x00000040 HasItem = 0x00000020 GetItemsUnconditionally = 0x00000080, GetItems = 0x00000100 CheckOut = 0x00000200, CheckIn = 0x00000400 UndoCheckOut = 0x00000800 IsCheckOutUser = 0x00001000

Repository

Prototype	`IRepository Repository{get; set;}`
Definition	IRepostiroy 인터페이스를 구현한 객체를 가져옵니다.

CreateItems

Prototype	`string[] CreateItems(object session, string[] itemNames, string[] descriptions, Stream[] items, string comment, OZErrorCode[] errCodes, string[] errMsgs)`	
Definition	새로운 아이템을 생성하고 생성된 아이템의 ID를 반환합니다.	
Argument	*session*	세션 ID
	itemNames	새로 생성할 아이템의 ID
	descriptions	새로 생성할 아이템의 설명 내용
	items	새로 생성할 아이템의 입력 스트림
	comment	주석문
	errCodes	에러 코드
	errMsgs	에러 메시지

ModifyItemName

Prototype	`string ModifyItemName(object session, string itemID, string newItemName, string comment)`

Definition	지정한 아이템 ID에 해당하는 아이템 이름을 변경하고 변경된 아이템 ID를 반환합니다.	
Argument	*session*	세션 ID
	itemID	이름을 변경할 아이템 ID
	newItemName	변경할 아이템 이름
	comment	주석문

■ DeleteItems

Prototype	`bool[] DeleteItems(object session, string[] itemIDs, bool[] toBeDestroyed, string comment, OZErrorCode[] errCodes, string[] errMsgs)`	
Definition	지정한 아이템을 리파지토리에서 삭제하고 아이템 삭제 성공 여부를 가져옵니다.	
Argument	*session*	세션 ID
	itemIDs	삭제할 아이템의 ID
	toBeDestroyed	아이템을 영구 삭제할 지 여부
	comment	주석문
	errCodes	에러 코드
	errMsgs	에러 메시지

■ UnDeleteItems

Prototype	`bool[] UndeleteItems(object session, string[] itemIDs, string comment, OZErrorCode[] errCodes, string[] errMsgs)`	
Definition	삭제된 아이템을 복원하고 복원 성공 여부를 가져옵니다. ※ 참고사항 : 아이템 삭제 시 "toBeDestroyed =false"로 설정하였을 경우 삭제된 아이템의 복원이 가능합니다.	
Argument	*session*	세션 ID
	itemIDs	복원할 아이템 ID
	comment	주석문
	errCodes	에러 코드
	errMsgs	에러 메시지

■ ModifyItemDescription

Prototype	`bool ModifyItemDescription(object session, string itemID, string description)`	
Definition	지정한 아이템 ID에 해당하는 아이템 설명을 변경하고 변경 성공 여부를 가져옵니다.	
Argument	*session*	세션 ID

	itemID	변경할 아이템 ID
	description	변경할 내용

■ **GetItemInfo**

Prototype	IItemInfo GetItemInfo(object session, string itemID)	
Definition	지정한 아이템 ID에 해당하는 아이템 정보를 가져옵니다.	
Argument	*session*	세션 ID
	itemID	아이템 정보를 가져올 아이템 ID

■ **HasTheItem**

Prototype	bool HasItem(object session, string itemID)	
Definition	지정한 아이템의 존재 여부를 가져옵니다.	
Argument	*session*	세션 ID
	itemID	아이템 ID

■ **GetItems**

Prototype	Stream[] GetItems(object session, string[] itemIDs, OZErrorCode[] errCodes, string[] errMsgs) Stream[] GetItems(object session, string[] itemIDs, long[] modifiedTimes, ItemOptions[] options, OZErrorCode[] errCodes, string[] errMsgs)	
Definition	지정한 아이템을 가져온 후 아이템 스트림을 가져옵니다.	
Argument	*session*	세션 ID
	itemIDs	가져올 아이템 ID
	modifiedTimes	클라이언트 시간
	options	에러 메시지 처리에 대한 옵션
	errCodes	에러 코드
	errMsgs	에러 메시지

■ **CheckOut**

Prototype	Stream[] CheckOut(object session, string[] itemIDs, string[] localCheckOutFolders, long[] localFileTimes, OZErrorCode[] errCodes, string[] errMsgs)	
Definition	지정한 아이템을 지정한 사용자 ID로 체크 아웃할 폴더에 체크 아웃을 하고 로컬 파일의 시간이 아이템 시간 보다 작을 경우 아이템 스트림을 가져옵니다.	
Argument	*session*	세션 ID

	itemIDs	체크 아웃할 아이템의 ID
	localCheckOutFolders	체크 아웃할 로컬 폴더 이름
	localFileTimes	체크 아웃할 아이템의 로컬 파일 시간
	errCodes	에러 코드
	errMsgs	에러 메시지

■ CheckIn

Prototype	`bool[] CheckIn(object session, string[] itemIDs, Stream[] items, string comment, bool[] keepCheckOut, OZErrorCode[] errCodes, string[] errMsgs)`	
Definition	지정한 아이템을 지정한 사용자 ID로 체크인하고 체크인 성공 여부를 가져옵니다.	
Argument	*session*	세션 ID
	itemIDs	체크인할 아이템 ID
	items	체크인할 아이템 스트림
	comment	주석문
	keepCheckOut	체크 아웃 상태를 유지할지 여부
	errCodes	에러 코드
	errMsgs	에러 메시지

■ UndoCheckOut

Prototype	`Stream[] UndoCheckOut(object session, string[] itemIDs, bool[] replace, OZErrorCode[] errCodes, string[] errMsgs)`	
Definition	지정한 아이템을 체크 아웃 취소합니다.	
Argument	*session*	세션 ID
	itemIDs	체크 아웃 취소할 아이템의 ID
	replaces	로컬 파일 변경 여부
	errCodes	에러 코드
	errMsgs	에러 메시지

■ IsCheckOutUser

Prototype	`bool[] IsCheckOutUser(object session, string[] itemIDs)`	
Definition	지정한 사용자가 지정한 아이템을 체크 아웃했는지 여부를 확인합니다	
Argument	*session*	세션 ID
	itemIDs	아이템 ID

/ / IRepositoryHistory

■ AccessType

Prototype	`HistoryAccess AccessType{get;}`
Definition	아이템 히스토리 인터페이스에 접근 가능한 메소드를 숫자값으로 가져옵니다. None = 0x00000000 All = 0x7FFFFFFF RollBack = 0x00000001 GetItem = 0x00000002 GetHistoryInfos = 0x00000004 GetDeletedItemHistoryInfos = 0x00000008 RemoveHistory = 0x00000010

■ Repository

Prototype	`IRepository Repository{get; set;}`
Definition	IRepostiroy 인터페이스를 구현한 객체를 가져옵니다.

■ RollBack

Prototype	`bool RollBack(object session, string itemID, int version, string comment)`	
Definition	지정한 아이템을 지정한 버전으로 복원시킵니다.	
Argument	*session*	세션 ID
	iItemID	복원시킬 아이템의 ID
	version	복원시킬 버전
	comment	주석문

■ GetItem

Prototype	`Stream GetItem(object session, string itemID, int version)`	
Definition	지정한 아이템 ID에 해당하는 아이템 중 지정한 버전의 아이템을 가져옵니다.	
Argument	*session*	세션 ID
	itemID	가져올 아이템의 ID
	version	가져올 아이템의 버전

■ GetHistoryInfos

Prototype	`IHistoryInfo[] GetHistoryInfos(object session, string itemID)`
Definition	지정한 아이템의 히스토리 정보를 가져옵니다..

Argument	*session*	세션 ID
	itemID	히스토리 정보를 가져올 아이템의 ID

■ **GetDeleteHistoryItemInfo**

Prototype	`IHistoryInfo[] GetDeletedItemHistoryInfos(object session, string itemID)`
Definition	삭제된 히스토리 리스트를 가져옵니다. ※ 참고사항 : "toBeDestroyed =false"로 설정하여 삭제한 아이템의 정보만 가져옵니다.

Argument	*session*	세션 ID
	itemIDs	히스토리 정보를 가져올 아이템의 ID

■ **RemoveHistory**

Prototype	`bool RemoveHistory(object session, string itemID, int version)`
Definition	지정한 아이템에 대해 특정 버전의 히스토리를 삭제합니다.

Argument	*session*	세션 ID
	itemID	히스토리를 삭제할 아이템의 ID
	version	히스토리를 삭제할 버전

// IRepositoryGroup

■ **AccessType**

Prototype	`GroupAccess AccessType{get;}`
Definition	그룹 인터페이스에 접근 가능한 메소드를 숫자값으로 가져옵니다. None = 0x00000000 All = 0x7FFFFFFF CreateGroup = 0x00000001 ModifyGroupName = 0x00000002 ModifyGroupDescription = 0x00000004 DeleteGroup = 0x00000008, CreateUser = 0x00000010 TransferUser = 0x00000020 GetUserInfos = 0x00000040 GetGroupInfo = 0x00000080 GetSubGroupInfos = 0x00000100 GetParentGroupInfo = 0x00000200 GetGroupID = 0x00000400 AddGroupAdministrator = 0x00000800 RemoveGroupAdministrator = 0x00001000 IsGroupAdministrator = 0x00002000

GetAdministrators = 0x00004000
TransferGroup = 0x00008000

■ **Repository**

Prototype	`IRepository Repository{get; set;}`
Definition	IRepostiroy 인터페이스를 구현한 객체를 가져옵니다.

■ **CreateGroup**

Prototype	`string CreateGroup(object session, string groupName, string parentGroupID, string description)`	
Definition	새로운 그룹을 생성하고 생성된 그룹의 ID를 반환합니다.	
Argument	*session*	세션 ID
	groupName	새로 생성할 그룹의 이름
	parentGroupID	새로 생성할 그룹의 상위 그룹 ID
	description	그룹에 대한 설명 내용

■ **ModifyGroupName**

Prototype	`string ModifyGroupName(object session, string groupID, string groupName)`	
Definition	지정한 그룹 ID에 해당하는 그룹의 이름을 변경하고 변경된 그룹 ID를 반환합니다.	
Argument	*session*	세션 ID
	groupID	그룹의 이름을 변경할 그룹 ID
	groupName	변경할 그룹의 이름

■ **ModifyGroupDescription**

Prototype	`bool ModifyGroupDescription(object session, string groupID, string description)`	
Definition	지정한 그룹 ID에 해당하는 그룹의 설명을 변경하고 변경 성공 여부를 반환합니다.	
Argument	*session*	세션 ID
	groupID	그룹의 설명을 변경할 그룹 ID
	description	변경할 그룹의 설명

■ **DeleteGroup**

Prototype	`bool DeleteGroup(object session, string groupID)`	
Definition	지정한 그룹 ID에 해당하는 그룹을 삭제하고 삭제 성공 여부를 반환합니다.	
Argument	*session*	세션 ID
	groupID	삭제할 그룹 ID

■ **CreateUser**

Prototype	`string CreateUser(object session, string userName, string password, string groupID, string description)`	
Definition	지정한 그룹에 새로운 사용자 ID를 생성하고 생성된 사용자 ID를 반환합니다.	
Argument	*session*	세션 ID
	userName	사용자 이름
	password	패스워드
	groupID	그룹ID
	description	사용자에 대한 설명 내용

■ **GetUserInfos**

Prototype	`IUserInfo[] GetUserInfos(object session, string groupID)`	
Definition	지정한 그룹 ID에 등록되어 있는 모든 사용자의 정보를 가져옵니다.	
Argument	*session*	세션 ID
	groupID	사용자의 정보를 가져올 그룹 ID

■ **GetGroupInfo**

Prototype	`IGroupInfo GetGroupInfo(object session, string groupID)`	
Definition	지정한 그룹 ID에 해당하는 그룹의 정보를 가져옵니다.	
Argument	*session*	세션 ID
	groupID	그룹 ID

■ **GetSubGroupInfos**

Prototype	`IGroupInfo[] GetSubGroupInfos(object session, string groupID)`	
Definition	지정한 그룹의 하위 그룹 정보를 가져옵니다.	
Argument	*session*	세션 ID
	groupID	하위 그룹의 정보를 가져올 그룹 ID

■ **GetParentGroupInfo**

Prototype	`IGroupInfo GetParentGroupInfo(object session, string groupID)`	
Definition	지정한 그룹의 상위 그룹의 정보를 가져옵니다.	
Argument	*session*	세션 ID
	groupID	상위 그룹의 정보를 가져올 그룹 ID

■ **GetGroupID**

Prototype	`string GetGroupID(object session, string userID)`	
Definition	지정한 사용자가 속해 있는 그룹의 정보를 가져옵니다.	
Argument	*session*	세션 ID
	userID	그룹 정보를 가져올 사용자 ID

■ **AddGroupAdministrator**

Prototype	`bool AddGroupAdministrator(object session, string userID, string groupID)`	
Definition	지정한 그룹의 그룹 관리자를 추가하고 추가 성공 여부를 반환합니다.	
Argument	*session*	세션 ID
	userID	그룹 관리자로 추가할 사용자 ID
	groupID	그룹 관리자를 추가할 그룹 ID

■ **RemoveGroupAdministrator**

Prototype	`bool RemoveGroupAdministrator(object session, string userID, string groupID)`	
Definition	지정한 그룹의 그룹 관리자를 해제하고 해제 성공 여부를 반환합니다.	
Argument	*session*	세션 ID
	userID	그룹 관리자 권한을 해제할 사용자 ID
	groupID	그룹 관리자를 해제할 그룹 ID

■ **IsGroupAdministrator**

Prototype	`bool IsGroupAdministrator(object session, string userID, string groupID)`	
Definition	지정한 사용자 ID에 해당하는 사용자가 해당 그룹의 관리자인지 여부를 확인합니다.	
Argument	*session*	세션 ID
	userID	관리자인지 체크할 사용자 ID

	groupID	그룹 ID

■ **GetAdministrators**

Prototype	`IUserInfo[] GetAdministrators(object session, string groupID)`	
Definition	지정한 그룹의 그룹 관리자 정보를 가져옵니다.	
Argument	*session*	세션 ID
	groupID	그룹 관리자 정보를 가져올 그룹 ID

■ **TransferUser**

Prototype	`bool TransferUser(object session, string userID, string newGroupID)`	
Definition	지정한 사용자의 그룹을 이동하고 그룹 이동 성공 여부를 가져옵니다.	
Argument	*session*	세션 ID
	userID	그룹을 이동할 사용자 ID
	newGroupID	이동할 그룹 ID

■ **TransferGroup**

Prototype	`bool TransferGroup(object session, string groupID, string targetGroupID)`	
Definition	지정한 그룹을 다른 그룹으로 이동하고 그룹 이동 성공 여부를 가져옵니다.	
Argument	*session*	세션 ID
	userID	그룹을 이동할 그룹 ID
	targetGroupID	이동할 그룹 ID

// IRepositoryUser

■ **AccessType**

Prototype	`UserAccess AccessType{get;}`

Definition	사용자 인터페이스에 접근 가능한 메소드를 숫자값으로 가져옵니다. None = 0x00000000 All = 0x7FFFFFFF CreateUser = 0x00000001 ModifyName = 0x00000002 ModifyPassword = 0x00000004 ModifyDescription = 0x00000008 DeleteUser = 0x00000010 GetUserInfo = 0x00000020 CheckPassword = 0x00000040 GetUserInfos = 0x00000080 IsAdministrator = 0x00000100

■ Repository

Prototype	`IRepository Repository{get; set;}`
Definition	IRepostiroy 인터페이스를 구현한 객체를 가져옵니다.

■ CreateUser

Prototype	`string CreateUser(object session, string userName, string password, string description)`	
Definition	새로운 사용자를 생성하고 생성된 사용자의 ID를 반환합니다.	
Argument	*session*	세션 ID
	userName	사용자 이름
	password	패스워드
	description	사용자에 대한 설명 내용

■ ModifyName

Prototype	`string ModifyName(object session, string userID, string newUserName)`	
Definition	지정한 사용자 ID에 해당하는 사용자의 이름을 변경하고 사용자 이름이 변경된 사용자 ID를 반환합니다.	
Argument	*session*	세션 ID
	userID	사용자 ID
	newUserName	변경할 사용자 이름

■ ModifyPassword

Prototype	`bool ModifyPassword(object session, string userID, string oldPassword, string newPassword)`

Definition	지정한 사용자의 패스워드를 변경하고 변경 성공 여부를 반환합니다.	
Argument	*session*	세션 ID
	userID	패스워드를 변경할 사용자 ID
	oldPassword	변경 전 패스워드
	newPpassword	변경할 패스워드

■ ModifyUserDescription

Prototype	`bool ModifyDescription(object session, string userID, string description)`
Definition	지정한 사용자 ID에 해당하는 사용자의 사용자 설명을 변경하고 변경 성공 여부를 반환합니다.
Argument	*session* 세션 ID
	userID 사용자 설명 내용을 변경할 사용자 ID
	description 변경할 사용자 설명

■ DeleteUser

Prototype	`bool DeleteUser(object session, string userID)`
Definition	지정한 사용자 ID에 해당하는 사용자의 모든 정보를 삭제하고 삭제 성공 여부를 반환합니다.
Argument	*session* 세션 ID
	userID 삭제할 사용자의 ID

■ GetUserInfo

Prototype	`IUserInfo GetUserInfo(object session, string userID)`
Definition	지정한 사용자 ID에 해당하는 사용자의 사용자 정보를 가져옵니다.
Argument	*session* 세션 ID
	userID 사용자 정보를 가져올 사용자 ID

■ CheckPassword

Prototype	`bool CheckPassword(object session, string userID, string password)`
Definition	사용자의 패스워드가 맞는지 여부를 체크합니다.
Argument	*session* 세션 ID
	userID 패스워드를 체크할 사용자 ID
	password 체크할 패스워드

■ **GetUserInfos**

Prototype	`IUserInfo[] GetUserInfos(object session)`	
Definition	모든 사용자의 사용자 정보를 가져옵니다.	
Argument	*session*	세션 ID

■ **IsAdministrator**

Prototype	`bool IsAdministrator(object session, string userID)`	
Definition	지정한 사용자 ID에 해당하는 사용자가 리파지토리의 관리자인지 여부를 확인합니다.	
Argument	*session*	세션 ID
	userID	관리자인지 체크할 사용자 ID

// IRepositoryMultiLoginUser

■ **AccessType**

Prototype	`MultiLoginAccess AccessType{get;}`
Definition	멀티로그인 인터페이스에 접근 가능한 메소드를 숫자값으로 가져옵니다. int ACCESS_MULTILOGINUSER_NOT = 0x00000000 int ACCESS_DISABLE_USER_LOGIN = 0x00000001 int ACCESS_ENABLE_USER_LOGIN = 0x00000002 int ACCESS_IS_LOGIN_ENABLED = 0x00000004

■ **Repository**

Prototype	`IRepository Repository{get; set;}`
Definition	IRepostiroy 인터페이스를 구현한 객체를 가져옵니다.

■ **DisableLogin**

Prototype	`void DisableLogin(object session, string userID)`	
Definition	지정한 세션 ID의 사용자를 로그인하지 못하도록 설정합니다.	
Argument	*session*	세션 ID
	userID	로그인하지 못하도록 할 사용자 ID

■ **EnableUserLogin**

Prototype	`void EnableLogin(object session, string userID)`	
Definition	지정한 세션 ID의 사용자를 로그인 가능하도록 설정합니다.	
Argument	*session*	세션 ID
	userID	로그인 가능하도록 할 사용자 ID

■ isLoginEnabled

Prototype	`bool IsLoginEnabled(object session, string userID)`	
Definition	지정한 세션 ID의 사용자가 로그인이 가능한지 여부를 가져옵니다.	
Argument	*session*	세션 ID
	userID	로그인 가능 가능 여부를 체크할 사용자 ID

/ / IRepositoryGroupAuthorityToCategory

■ AccessType

Prototype	`AuthorityAccess AccessType{get;}`
Definition	카테고리에 대한 그룹 권한 인터페이스에 접근 가능한 메소드를 숫자값으로 가져옵니다. None = 0x00000000 All = 0x7FFFFFFF ModifyGroupAuthToCategory = 0x00000001 GetGroupAuthToCategory = 0x00000002 GetGroupInfosOfCategory = 0x00000004 GetItemInfosInCategoryOfGroup = 0x00000008 GetCategoryInfosOfGroup = 0x00010000

■ Repository

Prototype	`IRepository Repository{get; set;}`
Definition	IRepostiroy 인터페이스를 구현한 객체를 가져옵니다.

■ Modify

Prototype	`bool Modify(object session, string categoryID, string groupID, Authority permission)`	
Definition	지정한 카테고리와 그룹의 권한 정보를 변경하고 변경 성공 여부를 가져옵니다.	
Argument	*session*	세션 ID
	categoryID	권한 정보를 변경할 카테고리 ID
	groupID	권한 정보를 변경할 그룹 ID
	permission	권한 정보 • 1 : VIEW • 2 : READ • 4 : WRITE ※ 참고사항 : 권한 정보는 OR 조건으로 처리해야합니다.

■ GetAuthority

Prototype	`Authority GetAuthority(object session, string groupID, string categoryID)`	
Definition	지정한 카테고리에 대한 지정한 그룹의 그룹 권한을 가져옵니다.	
Argument	*session*	세션 ID
	groupID	그룹 ID
	categoryID	권한 정보를 가져올 카테고리 ID

■ **GetGroupInfos**

Prototype	`IGroupInfo[] GetGroupInfos(object session, string categoryID, Authority permission)`	
Definition	지정한 카테고리 ID에 대해 permission 이상의 권한을 가지고 있는 모든 사용자 정보를 가져옵니다.	
Argument	*session*	세션 ID
	categoryID	카테고리 ID
	permission	권한 정보

■ **GetItemInfos**

Prototype	`IItemInfo[] GetItemInfos(object session, string groupID, string categoryID, Authority permission)`	
Definition	지정한 카테고리의 아이템 중에 지정한 그룹의 권한이 permission 이상인 모든 아이템의 목록을 가져옵니다.	
Argument	*session*	세션 ID
	groupID	그룹 ID
	categoryID	카테고리 ID
	permission	권한 정보

■ **GetCategoryInfos**

Prototype	`ICategoryInfo[] GetCategoryInfos(object session, string groupID, string categoryID, Authority permission)`	
Definition	지정한 카테고리의 하위 카테고리 중에 지정한 그룹의 권한이 permission 이상인 모든 카테고리의 목록을 가져옵니다.	
Argument	*session*	세션 ID
	groupID	그룹 ID
	categoryID	카테고리 ID
	permission	권한 정보

// IRepositoryGroupAuthorityToItem

AccessType

Prototype	`AuthorityAccess AccessType{get;}`
Definition	아이템에 대한 그룹 권한 인터페이스에 접근 가능한 메소드를 숫자값으로 가져옵니다. None = 0x00000000 All = 0x7FFFFFFF ModifyGroupAuthToItem = 0x00000010 GetGroupAuthToItem = 0x00000020 GetGroupInfosOfItem = 0x00000040 GetItemInfosOfGroup = 0x00000080

Repository

Prototype	`IRepository Repository{get; set;}`
Definition	IRepostiroy 인터페이스를 구현한 객체를 가져옵니다.

Modify

Prototype	`bool Modify(object session, string groupID, string itemID, Authority permission)`	
Definition	지정한 아이템 ID에 대한 지정한 그룹 ID의 권한 정보를 변경하고 변경 성공 여부를 반환합니다.	
Argument	*session*	세션 ID
	groupID	그룹 ID
	itemID	아이템 ID
	permission	권한 정보

GetAuthority

Prototype	`Authority GetAuthority(object session, string groupID, string itemID)`	
Definition	지정한 아이템 ID에 대한 지정한 그룹 ID의 권한 정보를 가져옵니다.	
Argument	*session*	세션 ID
	groupID	그룹 ID
	itemID	아이템 ID

GetGroupInfos

Prototype	`IGroupInfo[] GetGroupInfos(object session, string itemID, Authority permission)`
Definition	지정한 아이템 ID에 대해 permission 이상의 권한이 부여된 모든 그룹 정보를 가져옵니다.

Argument	*session*	세션 ID
	itemID	아이템 ID
	permission	권한 정보

■ **GetItemInfos**

Prototype	`IItemInfo[] GetItemInfos(object session, string groupID, Authority permission)`
Definition	지정한 그룹 ID에 해당하는 그룹에게 permission 이상의 권한이 부여된 모든 아이템의 정보를 가져옵니다.
Argument	*session* 세션 ID
	groupID 그룹 ID
	permission 권한 정보

/ / IRepositoryUserAuthorityToCategory

■ **AccessType**

Prototype	`AuthorityAccess AccessType{get;}`
Definition	카테고리에 대한 사용자 권한 인터페이스에 접근 가능한 메소드를 숫자값으로 가져옵니다. None = 0x00000000 All = 0x7FFFFFFF ModifyUserAuthToCategory = 0x00000100 GetUserAuthToCategory = 0x00000200, GetUserInfosOfCategory = 0x00000400 GetItemInfosInCategoryOfUser = 0x00000800 GetCategoryInfosOfUser = 0x00020000

■ **Repository**

Prototype	`IRepository Repository{get; set;}`
Definition	OZRepostiroy 인터페이스를 구현한 클래스를 가져옵니다.

■ **Modify**

Prototype	`bool Modify(object session, string userID, string categoryID, Authority permission)`	
Definition	지정한 카테고리 ID에 대한 지정한 사용자 ID의 권한 정보를 변경하고 변경 성공 여부를 반환합니다.	
Argument	*session*	세션 ID
	userID	사용자 ID
	categoryID	카테고리 ID

	permission	권한 정보

■ **GetAuthority**

Prototype	`Authority GetAuthority(object session, string userID, string categoryID)`	
Definition	지정한 카테고리 ID에 대한 지정한 사용자 ID의 권한 정보를 가져옵니다.	
Argument	*session*	세션 ID
	userID	사용자 ID
	categoryID	카테고리 ID

■ **GetUserInfos**

Prototype	`IUserInfo[] GetUserInfos(object session, string categoryID, Authority permission)`	
Definition	지정한 카테고리 ID에 대해 permission 이상의 권한이 부여된 모든 사용자 정보를 가져옵니다.	
Argument	*session*	세션 ID
	categoryID	카테고리 ID
	permission	권한 정보

■ **GetItemInfos**

Prototype	`IItemInfo[] GetItemInfos(object session, string userID, string categoryID, Authority permission)`	
Definition	지정한 카테고리의 아이템 중에 지정한 사용자의 권한이 permission 이상인 모든 아이템의 목록을 가져옵니다.	
Argument	*session*	세션 ID
	userID	사용자 ID
	categoryID	카테고리 ID
	permission	권한 정보

■ **GetCategoryInfos**

Prototype	`ICategoryInfo[] GetCategoryInfos(object session, string userID, string categoryID, Authority permission)`	
Definition	지정한 카테고리의 하위 카테고리 중에 지정한 사용자의 권한이 permission 이상인 모든 카테고리의 목록을 가져옵니다.	
Argument	*session*	세션 ID
	userID	사용자 ID

	categoryID	카테고리 ID
	permission	권한 정보

// IRepositoryUserAuthorityItem

■ **AccessType**

Prototype	`AuthorityAccess AccessType{get;}`
Definition	아이템에 대한 사용자 권한 인터페이스에 접근 가능한 메소드를 숫자값으로 가져옵니다. None = 0x00000000 All = 0x7FFFFFFF ModifyUserAuthToItem = 0x00001000 GetUserAuthToItem = 0x00002000 GetUserInfosOfItem = 0x00004000 GetItemInfosOfUser = 0x00008000

■ **Repository**

Prototype	`IRepository Repository{get; set;}`
Definition	IRepostiroy 인터페이스를 구현한 객체를 가져옵니다.

■ **Modify**

Prototype	`bool Modify(object session, string userID, string itemID, Authority permission)`	
Definition	지정한 아이템에 대한 사용자의 권한 정보를 변경하고 변경 성공 여부를 가져옵니다.	
Argument	*session*	세션 ID
	userID	사용자 ID
	iItemID	아이템 ID
	permission	권한 정보

■ **GetAuthorigy**

Prototype	`Authority GetAuthorigy(object session, string userID, string itemID)`	
Definition	지정한 아이템에 대한 사용자의 권한 정보를 가져옵니다.	
Argument	*session*	세션 ID
	userID	사용자 ID
	itemID	권한 정보

■ **GetUserInfos**

Prototype	`IUserInfo[] GetUserInfos(object session, string itemID, Authority permission)`	
Definition	지정한 아이템에 대한 사용자의 권한이 permission 이상인 모든 아이템 정보를 가져옵니다.	
Argument	*session*	세션 ID
	itemID	아이템 ID
	permission	권한 정보

■ **GetItemInfos**

Prototype	`IItemInfo[] GetItemInfos(object session, string userID, Authority permission)`	
Definition	지정한 사용자의 권한이 permission 이상인 모든 아이템 정보를 가져옵니다.	
Argument	session	세션 ID
	userID	아이템 ID
	permission	권한 정보

// ICategoryInfo

■ **ID**

Prototype	`string ID{get;}`
Definition	현재 카테고리의 ID를 가져옵니다.

■ **Name**

Prototype	`string Name{get;}`
Definition	현재 카테고리의 이름을 가져옵니다.

■ **ParentID**

Prototype	`string ParentID{get;}`
Definition	현재 카테고리의 상위 카테고리 ID를 가져옵니다.

■ **ParentName**

Prototype	`string ParentName{get;}`
Definition	현재 카테고리의 상위 카테고리 이름을 가져옵니다.

■ **Comment**

Prototype	`string Comment{get;}`
Definition	현재 카테고리의 주석문을 가져옵니다.

■ **Description**

Prototype	`string Description{get;}`
Definition	현재 카테고리의 설명 내용을 가져옵니다.

■ **UpdateTime**

Prototype	`long UpdateTime{get;}`
Definition	현재 카테고리를 갱신한 시간 정보를 가져옵니다.

■ **CreatedUserID**

Prototype	`string CreatedUserID{get;}`
Definition	현재 카테고리를 생성한 사용자 ID를 가져옵니다.

■ **CreatedUserName**

Prototype	`string CreatedUserName{get;}`
Definition	현재 카테고리를 생성한 사용자 이름을 가져옵니다.

■ **Permission**

Prototype	`Authority Permission{get;}`
Definition	현재 카테고리의 권한 정보를 가져옵니다.

// IItemInfo

■ **IsCheckedOut**

Prototype	`bool IsCheckedOut{get;}`
Definition	해당 아이템의 체크 아웃 여부를 가져옵니다

■ **IsDeletable**

Prototype	`bool IsDeletable{get;}`
Definition	해당 아이템의 삭제 여부를 가져옵니다. ※ 참고사항 : "toBeDestroyed =false"로 설정하여 삭제한 아이템의 정보만 가져옵니다.

■ **ID**

Prototype	`string ID{get;}`
Definition	해당 아이템의 아이템 ID를 가져옵니다.

- **Name**

Prototype	`string Name{get;}`
Definition	해당 아이템의 아이템 이름을 가져옵니다.

- **CategoryID**

Prototype	`string CategoryID{get;}`
Definition	해당 아이템이 속한 카테고리의 카테고리 ID를 가져옵니다.

- **CategoryName**

Prototype	`string CategoryName{get;}`
Definition	해당 아이템이 속한 카테고리의 카테고리 이름을 가져옵니다.

- **UpdateTime**

Prototype	`long UpdateTime{get;}`
Definition	해당 아이템을 갱신한 시간 정보를 가져옵니다.

- **Description**

Prototype	`string Description{get;}`
Definition	해당 아이템의 설명 내용을 가져옵니다.

- **Comment**

Prototype	`string Comment{get;}`
Definition	해당 아이템의 주석문을 가져옵니다.

- **CreatedUserID**

Prototype	`string CreatedUserID{get;}`
Definition	해당 아이템을 생성한 사용자 ID를 가져옵니다.

- **CreatedUserName**

Prototype	`string CreatedUserName{get;}`
Definition	해당 아이템을 생성한 사용자 이름을 가져옵니다.

■ **CheckOutUser**

Prototype	`string CheckOutUser{get;}`
Definition	해당 아이템을 체크 아웃한 사용자 이름을 가져옵니다.

■ **CheckOutLocalPath**

Prototype	`string CheckOutLocalPath{get;}`
Definition	체크 아웃한 아이템의 폴더 이름을 가져옵니다.

■ **Permission**

Prototype	`Authority Permission{get;}`
Definition	해당 아이템의 권한 정보를 가져옵니다.

// IHistoryInfo

■ **ItemID**

Prototype	`string ItemID{get;}`
Definition	아이템 ID를 가져옵니다.

■ **ItemName**

Prototype	`string ItemName{get;}`
Definition	아이템 이름을 가져옵니다.

■ **Version**

Prototype	`int Version{get;}`
Definition	아이템의 버전 개수를 가져옵니다.

■ **CheckInTime**

Prototype	`long CheckInTime{get;}`
Definition	아이템의 버전 별 체크인 시간을 가져옵니다.

■ **CheckInUser**

Prototype	`string CheckInUser{get;}`
Definition	아이템의 버전 별 체크인 사용자 ID를 가져옵니다.

■ **CheckInComment**

Prototype	`string CheckInComment{get;}`
Definition	아이템의 버전 별 체크인 주석문를가져옵니다.

// IGroupInfo

■ **Administrators**

Prototype	`IDictionary Administrators{get;}`
Definition	해당 그룹의 관리자 정보를 가져옵니다. ※ 참고사항 : 관리자 정보는 "key=*사용자ID*, value=*사용자 이름*"으로 구성된 HashMap으로 반환됩니다.

■ **ID**

Prototype	`string ID{get;}`
Definition	해당 그룹의 그룹 ID를 가져옵니다.

■ **Name**

Prototype	`string Name{get;}`
Definition	해당 그룹의 그룹 이름을 가져옵니다.

■ **ParentID**

Prototype	`string ParentID{get;}`
Definition	해당 그룹의 상위 그룹 ID를 가져옵니다.

■ **ParentName**

Prototype	`string ParentName{get;}`
Definition	해당 그룹의 상위 그룹 이름을 가져옵니다.

■ **Description**

Prototype	`string Description{get;}`
Definition	해당 그룹의 설명 내용을 가져옵니다.

■ **UpdateTime**

Prototype	`long UpdateTime{get;}`
Definition	해당 그룹을 갱신한 시간 정보를 가져옵니다.

■ **Permission**

Prototype	Authority Permission{get;}
Definition	해당 그룹의 권한 정보를 가져옵니다.

// IUserInfo

■ **ID**

Prototype	string ID{get;}
Definition	해당 사용자의 사용자 ID를 가져옵니다.

■ **Name**

Prototype	string Name{get;}
Definition	해당 사용자의 사용자 이름을 가져옵니다.

■ **GroupID**

Prototype	string GroupID{get;}
Definition	해당 사용자가 속한 그룹의 그룹 ID를 가져옵니다.

■ **GroupName**

Prototype	string GroupName{get;}
Definition	해당 사용자가 속한 그룹의 그룹 이름을 가져옵니다.

■ **Description**

Prototype	string Description{get;}
Definition	해당 사용자의 설명 내용을 가져옵니다.

■ **UpdateTime**

Prototype	long UpdateTime{get;}
Definition	해당 사용자를 갱신한 시간 정보를 가져옵니다.

■ **IsLoggedIn**

Prototype	bool IsLoggedIn{get;}
Definition	해당 사용자의 로그인 여부를 가져옵니다.

■ **SessionID**

Prototype	string SessionID{get;}

Definition	해당 사용자가 로그인 시 생성된 세션 ID를 가져옵니다.

■ **IsLoginEnabled**

Prototype	bool IsLoginEnabled{get;}
Definition	해당 사용자의 로그인 가능 여부를 가져옵니다.

■ **LoginIP**

Prototype	string LoginIP{get;}
Definition	해당 사용자가 로그인한 PC의 IP 정보를 가져옵니다.

■ **LastLoginTime**

Prototype	long LastLoginTime{get;}
Definition	해당 사용자가 마지막으로 로그인한 시간 정보를 가져옵니다.

■ **Permission**

Prototype	Authority Permission{get;}
Definition	해당 사용자의 권한 정보를 가져옵니다.

// ILoginInfo

■ **ClientType**

Prototype	ClientType ClientType{get;}
Definition	클라이언트에서 호출한 클라이언트의 종류를 가져옵니다.

■ **MACAddress**

Prototype	string MACAddress{get;}
Definition	클라이언트로부터 MAC Address 정보를 가져옵니다.

■ **IPAddress**

Prototype	string IPAddress{get;}
Definition	클라이언트로부터 IP Address 정보를 가져옵니다.

■ **HDDSerialNo**

Prototype	string HDDSerialNO{get;}
Definition	클라이언트로부터 HDD Serial 번호를 가져옵니다.

■ **ProtocolVersion**

Prototype	`int ProtocolVersion{get;}`
Definition	클라이언트로부터 프로토콜 버전을 가져옵니다.

■ **UserID**

Prototype	`string UserID{get;}`
Definition	클라이언트가 로그인한 사용자 ID를 가져옵니다.

■ **UserName**

Prototype	`string UserName{get;}`
Definition	클라이언트가 로그인한 사용자 이름을 가져옵니다.

■ **SessionID**

Prototype	`string SessionID{get;}`
Definition	클라이언트가 로그인 시 생성된 세션 ID를 가져//옵니다.

■ **ClientIP**

Prototype	`string ClientIP{get;}`
Definition	클라이언트가 요청한 클라이언트 IP를 가져옵니다.

Class OZRepositoryException

// 패키지명 : oz.framework.repositoryex

■ **public class OZRepositoryException extends Exception**

■ **Method Summary**

- **public OZRepositoryException(String _msg)**
- **public OZRepositoryException(int code,String _msg)**
- **public OZErrorCode ErrorCode{get;}**
- **public string Message{get;}**

■ **Method Detail**

- OZRepositoryException

Prototype	public OZRepositoryException(String _msg) public OZRepositoryException(int code,String _msg)	
Definition	OZRepositoryException 생성자	
Argument	*code*	에러 코드
	_msg	에러 메시지

- ErrorCode

Prototype	public OZErrorCode ErrorCode{get;}
Definition	에러 코드

- Message

Prototype	public string Message{get;}
Definition	에러 메시지

Class RepositoryEX

oz.framework.api.RepositoryEx 클래스에 대한 Constructor 및 메소드는 다음과 같습니다.

Constructor Summary

- **public RepositoryEx(String ip, int port, String id, String pw) throws OZCPException**
- **public RepositoryEx(String ip, int port, String id, String pw, boolean useUSL) throws OZCPException**
- **public RepositoryEx(String iurl, String id, String pw) throws OZCPException**
- **public RepositoryEx(String iurl, String id, String pw, boolean useUSL) throws OZCPException**

Method Summary

- **public OZErrorCode GetLastErrorCode(int index)**
- **public string GetLastErrorMessage(int index)**
- **public void SetConfiguration(NameValueCollection conf)**
- **public NameValueCollection GetConfiguration()**
- **public void Reload()**
- **public string[] CreateItems(string[] itemNames, string[] descriptions, string[] categoryIDs, bool[] areCompressed, Stream[] items, string comment)**
- **public string ModifyItemName(string itemID, string newItemName, string comment)**
- **public bool[] DeleteItems(string[] itemIDs, bool[] toBeDestroyed, string comment)**
- **public bool[] UndeleteItems(string[] itemIDs, string comment)**
- **public bool ModifyItemDescription(string itemID, string description)**
- **public IItemInfo GetItemInfo(string itemID)**
- **public bool HasItem(string itemID)**

- public Stream[] GetItems(string[] itemIDs, bool[] areCompressed, bool[] needObjStreams)
- public Stream[] GetItems(string[] itemIDs, long[] modifiedTimes, bool[] areCompressed, bool[] needObjStreams)
- public Stream[] CheckOut(string[] itemIDs, string[] checkOutFolders, long[] localFileTimes, bool[] areCompressed)
- public bool[] CheckIn(string[] itemIDs, bool[] areCompressed, Stream[] items, string comment, bool[] keepCheckOut)
- public Stream[] UndoCheckOut(string[] itemIDs, bool[] replaceItems, bool[] areCompressed)
- public bool[] IsCheckOutUser(string[] itemIDs)
- public bool RollBack(string itemID, int version, string comment)
- public Stream GetItem(string itemID, int version, bool isCompressed)
- public IHistoryInfo[] GetHistoryInfos(string itemID)
- public IHistoryInfo[] GetDeletedItemHistoryInfos(string itemID)
- public bool RemoveHistory(string itemID, int version)
- public bool TransferItems(string[] itemIDs, string targetCategoryID)
- public bool TransferCategory(string categoryID, string targetCategoryID)
- public string[] CreateCategories(string[] categoryNames, string[] parentCategoryIDs, string comment)
- public string ModifyCategoryName(string categoryID, string newCategoryName, string comment)
- public bool[] DeleteCategories(string[] categoryIDs, bool[] toBeDestroyed, string comment)
- public bool[] UndeleteCategories(string[] categoryIDs, string comment)
- public int GetItemCount(string categoryID)
- public IItemInfo[] GetItemInfos(string categoryID)
- public ICategoryInfo[] GetCategoryInfos(string categoryID)
- public string GetCategoryID(string itemID)
- public ICategoryInfo GetCategoryInfo(string categoryID)
- public IItemInfo[] GetDeletedItemInfos(string categoryID)
- public IItemInfo[] SearchItem(string categoryID, SearchOption options, string criteria)
- public string CreateUser(string userName, string password, string description)
- public string ModifyUserName(string userID, string userName)

- public bool ModifyUserPassword(string userID, string oldPwd, string newPwd)
- public bool ModifyUserDescription(string userID, string description)
- public bool DeleteUser(string userID)
- public IUserInfo GetUserInfo(string userID)
- public bool CheckPassword(string userID, string password)
- public IUserInfo[] GetUserInfos()
- public void DisableLogin(string userID)
- public void EnableLogin(string userID)
- public bool IsLoginEnabled(string userID)
- public string CreateGroup(string groupName, string parentGroupID, string description)
- public string ModifyGroupName(string groupID, string groupName)
- public bool ModifyGroupDescription(string groupID, string description)
- public bool DeleteGroup(string groupID)
- public string CreateUser(string userName, string password, string groupID, string description)
- public bool TransferUser(string userID, string newGroupID)
- public bool TransferGroup(string groupID, string targetGroupID)
- public IUserInfo[] GetUserInfos(string groupID)
- public IGroupInfo GetGroupInfo(string groupID)
- public IGroupInfo[] GetSubGroupInfos(string groupID)
- public IGroupInfo GetParentGroupInfo(string groupID)
- public string GetGroupID(string userID)
- public bool AddGroupAdministrator(string userID, string groupID)
- public bool RemoveGroupAdministrator(string userID, string groupID)
- public bool IsGroupAdministrator(string userID, string groupID)
- public IUserInfo[] GetGroupAdministrators(string groupID)
- public bool IsAdministrator(string userID)
- public Stream Distribute(string categoryID, bool isRecursive)
- public bool Upload(string categoryID, Stream in)
- public bool ModifyUserAuthorityToItem(string userID, string itemID, Authority permission)
- public bool ModifyUserAuthorityToCategory(string userID, string categoryID, Authority permission)
- public bool ModifyGroupAuthorityToItem(string groupID, string itemID,

Authority permission)

- public bool ModifyGroupAuthorityToCategory(string groupID, string categoryID, Authority permission)
- public Authority GetUserAuthorityToItem(string userID, string itemID)
- public Authority GetUserAuthorityToCategory(string userID, string categoryID)
- public Authority GetGroupAuthorityToItem(string groupID, string itemID)
- public Authority GetGroupAuthorityToCategory(string groupID, string categoryID)
- public IGroupInfo[] GetGroupInfosOfItem(string itemID, Authority permission)
- public IGroupInfo[] GetGroupInfosOfCategory(string categoryID, Authority permission)
- public IUserInfo[] GetUserInfosOfItem(string itemID, Authority permission)
- public IUserInfo[] GetUserInfosOfCategory(string categoryID, Authority permission)
- public ICategoryInfo[] GetCategoryInfosOfUser(string userID, string categoryID, Authority permission)
- public ICategoryInfo[] GetCategoryInfosOfGroup(string groupID, string categoryID, Authority permission)
- public IItemInfo[] GetItemInfosOfUser(string userID, string categoryID, Authority permission)
- public IItemInfo[] GetItemInfosOfGroup(string groupID, string categoryID, Authority permission)

Constructor Detail

- RepositoryEx

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

Method Detail

■ GetLastErrorCode

Prototype	`public OZErrorCode GetLastErrorCode(int index)`
Definition	마지막에 호출한 메소드의 에러 코드를 가져옵니다.
Argument	*index* 메소드 호출 시 요청 인덱스

■ GetLastErrorMessage

Prototype	`public OZErrorCode GetLastErrorMessage(int index)`
Definition	마지막에 호출한 메소드의 에러 메시지를 가져옵니다.
Argument	*index* 메소드 호출 시 요청 인덱스

■ SetConfiguration

Prototype	`public void SetConfiguration(NameValueCollection conf)`
Definition	리파지토리 설정을 변경합니다.
Argument	*conf* 변경할 설정

■ **GetConfiguration**

Prototype	`public NameValueCollection GetConfiguration()`
Definition	리파지토리 설정을 가져옵니다.

■ **Reload**

Prototype	`public void Reload()`
Definition	리파지토리를 재시작합니다.

■ **CreateItems**

Prototype	`public string[] CreateItems(string[] itemNames, string[] descriptions, string[] categoryIDs, bool[] areCompressed, Stream[] items, string comment)`	
Definition	아이템을 생성하고 생성된 아이템 ID를 반환합니다.	
Argument	*itemNames*	새로 생성할 아이템의 이름
	descriptions	새로 생성할 아이템의 설명 내용
	categoryIDs	새로 생성할 아이템이 카테고리 이름
	areCompressed	압축 여부
	items	새로 생성할 아이템의 입력 스트림
	comment	주석문

■ **ModifyItemName**

Prototype	`public string ModifyItemName(string itemID, string newItemName, string comment)`	
Definition	아이템 이름을 변경하고 변경된 아이템 ID를 반환합니다.	
Argument	*itemID*	이름을 변경할 아이템 ID
	newItemName	변경할 아이템 이름
	comment	주석문

■ **DeleteItems**

Prototype	`public bool[] DeleteItems(string[] itemIDs, bool[] toBeDestroyed, string comment)`

Definition	지정한 아이템을 리파지토리에서 삭제하고 아이템 삭제 성공 여부를 가져옵니다.	
Argument	*itemIDs*	삭제할 아이템 ID
	toBeDestroyed	아이템을 영구 삭제할 지 여부
	comment	주석문

■ UnDeleteItems

Prototype	`public bool[] UndeleteItems(string[] itemIDs, string comment)`	
Definition	삭제된 아이템을 복원하고 복원 성공 여부를 가져옵니다. ※ 참고사항 : 아이템 삭제 시 "toBeDestroyed=false"로 설정하였을 경우 삭제된 아이템의 복원이 가능합니다.	
Argument	*itemIDs*	복원할 아이템 ID
	comment	주석문

■ ModifyItemDescription

Prototype	`public bool ModifyItemDescription(string itemID, string description)`	
Definition	지정한 아이템 ID에 해당하는 아이템의 설명을 변경하고 변경 성공 여부를 반환합니다.	
Argument	*itemID*	설명을 변경할 아이템의 ID
	description	아이템 설명 내용

■ GetItemInfo

Prototype	`public IItemInfo GetItemInfo(string itemID)`	
Definition	지정한 아이템 ID에 해당하는 아이템의 정보를 가져옵니다.	
Argument	*itemID*	정보를 가져올 아이템 ID

■ hasTheItem

Prototype	`public bool HasItem(string itemID)`	
Definition	지정한 아이템의 존재 여부를 확인합니다.	
Argument	*itemID*	체크할 아이템의 이름

■ GetItems

Prototype	`public Stream[] GetItems(string[] itemIDs, bool[] areCompressed, bool[] needObjStreams)`

Definition	지정한 아이템ID에 해당하는 아이템을 가져옵니다.	
Argument	*itemIDs*	가져올 아이템 ID
	areCompressed	압축 여부
	needObjStreams	ODI 객체 스트림 필요 여부

■ **GetItems**

Prototype	public Stream[] GetItems(string[] itemIDs, long[] modifiedTimes, bool[] areCompressed, bool[] needObjStreams)	
Definition	지정한 아이템ID에 해당하는 아이템을 가져옵니다.	
Argument	*itemIDs*	가져올 아이템 ID
	modifiedTimes	압축 여부
	areCompressed	아이템을 최종 변경한 시간
	needObjStreams	ODI 객체 스트림 필요 여부

■ **CheckOut**

Prototype	public Stream[] CheckOut(string[] itemIDs, string[] checkOutFolders, long[] localFileTimes, bool[] areCompressed)	
Definition	지정한 아이템을 체크 아웃할 폴더에 체크 아웃합니다.	
Argument	*itemIDs*	체크 아웃할 아이템의 ID
	checkOutFolders	체크 아웃할 폴더 이름
	localFileTimes	체크 아웃할 아이템의 로컬 파일 시간
	areCompressed	압축 여부

■ **checkIn**

Prototype	public bool[] CheckIn(string[] itemIDs, bool[] areCompressed, Stream[] items, string comment, bool[] keepCheckOut)	
Definition	지정한 아이템을 체크인합니다.	
Argument	*itemIDs*	체크인할 아이템의 ID
	areCompressed	압축 여부
	items	아이템의 입력 스트림
	comment	주석문
	keepCheckOut	체크 아웃 상태를 유지할지 여부

■ **UndoCheckOut**

Prototype	public Stream[] UndoCheckOut(string[] itemIDs, bool[] replaceItems, bool[] areCompressed)	
Definition	지정한 아이템을 체크 아웃 취소합니다.	
Argument	*itemIDs*	체크 아웃 취소할 아이템의 ID
	replaceItems	로컬 작업 폴더의 아이템을 가져올지 여부
	areCompressed	압축 여부

■ **IsCheckOutUser**

Prototype	public boolean[] isCheckOutUser(String[] ItemIDs) throws OZCPException	
Definition	현재 사용자가 지정한 아이템을 체크 아웃했는지 여부를 확인합니다.	
Argument	*itemIDs*	아이템 ID

■ **RollBack**

Prototype	public bool RollBack(string itemID, int version, string comment)	
Definition	지정한 아이템을 지정한 버전으로 복원시킵니다.	
Argument	*itemID*	복원시킬 아이템의 ID
	version	복원시킬 버전
	comment	주석문

■ **GetItem**

Prototype	public Stream GetItem(string itemID, int version, bool isCompressed)	
Definition	지정한 아이템 ID에 해당하는 아이템 중 지정한 버전의 아이템을 가져옵니다.	
Argument	*itemID*	가져올 아이템의 ID
	version	가져올 아이템의 버전
	isCompressed	압축 여부

■ **GetHistoryItems**

Prototype	public IHistoryInfo[] GetHistoryInfos(string itemID)	
Definition	지정한 아이템의 히스토리 정보를 가져옵니다.	
Argument	*itemID*	히스토리 정보를 가져올 아이템의 ID

■ **GetDeletedItemHistoryInfos**

Prototype	public IHistoryInfo[] GetDeletedItemHistoryInfos(string itemID)
Definition	해당 삭제된 아이템의 정보를 가져옵니다. ※ 참고사항 : "toBeDestroyed=false"로 설정하여 삭제한 아이템의 정보만 가져옵니다.
Argument	*itemID* 정보를 가져올 아이템의 ID

■ **RemoveHistory**

Prototype	public bool RemoveHistory(string itemID, int version)	
Definition	지정한 아이템에 대해 특정 버전의 히스토리를 삭제합니다.	
Argument	*itemID*	히스토리를 삭제할 아이템의 ID
	version	히스토리를 삭제할 버전

■ **CreateCategories**

Prototype	public string[] CreateCategories(string[] categoryNames, string[] parentCategoryIDs, string comment)	
Definition	카테고리를 새로 생성하고 생성된 카테고리의 ID를 반환합니다.	
Argument	*categoryName*	생성할 카테고리의 이름
	parentCategoryIDs	생성할 카테고리의 상위 카테고리 ID
	comment	주석문

■ **ModifyCategoryName**

Prototype	public string ModifyCategoryName(string categoryID, string newCategoryName, string comment)	
Definition	지정한 카테고리 ID에 해당하는 카테고리의 이름을 변경합니다.	
Argument	*categoryID*	이름을 변경할 카테고리의 ID
	newCategoryName	변경할 카테고리 이름
	comment	주석문

■ **DeleteCategories**

Prototype	public bool[] DeleteCategories(string[] categoryIDs, bool[] toBeDestroyed, string comment)	
Definition	지정한 카테고리 ID에 해당하는 카테고리를 삭제합니다.	
Argument	*categoryIDs*	삭제할 카테고리의 ID
	toBeDestroyed	카테고리를 영구 삭제할 지 여부
	comment	주석문

■ **unDeleteCategories**

Prototype	`public bool[] UndeleteCategories(string[] categoryIDs, string comment)`	
Definition	삭제된 카테고리를 복원하고 복원 성공 여부를 가져옵니다. ※ 참고사항 : 카테고리 삭제 시 "toBeDestroyed=false"로 설정하였을 경우 삭제된 카테고리의 복원가 가능합니다.	
Argument	*categoryIDs*	삭제된 카테고리의 ID
	comment	주석문

■ **GetItemCount**

Prototype	`public int GetItemCount(string categoryID)`	
Definition	지정한 카테고리에 속해 있는 모든 아이템의 개수를 가져옵니다.	
Argument	*categoryID*	아이템의 개수를 얻을 카테고리의 ID

■ **GetItemInfos**

Prototype	`public IItemInfo[] GetItemInfos(string categoryID)`	
Definition	지정한 카테고리에 속해 있는 모든 아이템 정보를 가져옵니다.	
Argument	*categoryID*	아이템 정보를 얻을 카테고리 ID

■ **GetCategoryInfos**

Prototype	`public ICategoryInfo[] GetCategoryInfos(string categoryID)`	
Definition	지정한 카테고리에 속해 있는 모든 하위 카테고리 정보를 가져옵니다.	
Argument	*categoryID*	아이템의 개수를 얻을 카테고리의 ID

■ **GetCategoryID**

Prototype	`public string GetCategoryID(string itemID)`	
Definition	지정한 아이템이 존재하는 카테고리의 ID를 반환합니다.	
Argument	*itemID*	카테고리의 ID를 얻을 아이템 ID

■ **GetCategoryInfo**

Prototype	`public ICategoryInfo GetCategoryInfo(string categoryID)`	
Definition	지정한 카테고리의 정보를 가져옵니다..	
Argument	*categoryID*	정보를 가져올 카테고리 ID

■ **GetDeletedItemInfos**

Prototype	`public IItemInfo[] GetDeletedItemInfos(string categoryID)`

Definition	지정한 카테고리의 삭제된 아이템 정보를 가져옵니다. ※ 참고사항 : "toBeDestroyed=false"로 설정하여 삭제한 아이템의 정보만 가져옵니다.
Argument	*categoryID* 삭제된 아이템 정보를 가져올 카테고리 ID

■ TransferCategory

Prototype	`public bool TransferCategory(string categoryID, string targetCategoryID)`
Definition	지정한 카테고리를 다른 카테고리로 이동하고 카테고리 이동 성공 여부를 가져옵니다.
Argument	*categoryID* 카테고리를 이동할 카테고리 ID
	targetCategoryID 이동할 카테고리 ID

■ TransferItems

Prototype	`public bool TransferItems(string[] itemIDs, string targetCategoryID)`
Definition	지정한 아이템의 카테고리를 이동하고 카테고리 이동 성공 여부를 가져옵니다
Argument	*itemIDs* 카테고리를 이동할 아이템 ID
	targetCategoryID 이동할 카테고리 ID

■ SearchItem

Prototype	`public IItemInfo[] SearchItem(string categoryID, SearchOption options, string criteria)`
Definition	지정한 카테고리의 아이템 중 지정한 옵션 및 조회 조건에 해당하는 아이템을 가져옵니다
Argument	*categoryID* 카테고리를 이동할 아이템 ID
	options 조회 옵션
	criteria 조회 조건

■ CreateUser

Prototype	`public string CreateUser(string userName, string password, string description)`
Definition	새로운 사용자를 생성하고 생성된 사용자의 ID를 반환합니다.
Argument	*userName* 사용자 이름
	password 패스워드
	description 사용자에 대한 설명 내용

■ **ModifyUserName**

Prototype	public string ModifyUserName(string userID, string userName)	
Definition	지정한 사용자 ID에 해당하는 사용자의 이름을 변경하고 변경 성공 여부를 반환합니다.	
Argument	*userID*	사용자 ID
	userName	변경할 사용자 이름

■ **ModifyUserPassword**

Prototype	public bool ModifyUserPassword(string userID, string oldPwd, string newPwd)	
Definition	지정한 사용자 ID에 해당하는 사용자의 패스워드를 변경하고 변경 성공 여부를 반환합니다.	
Argument	*userID*	사용자 ID
	oldPwd	변경 전 패스워드
	newPwd	변경 후 패스워드

■ **ModifyUserDescription**

Prototype	public bool ModifyUserDescription(string userID, string description)	
Definition	지정한 사용자 ID에 해당하는 사용자의 설명 내용을 변경하고 변경 성공 여부를 반환합니다.	
Argument	*userID*	사용자 ID
	description	변경할 설명 내용

■ **DeleteUser**

Prototype	public bool DeleteUser(string userID)	
Definition	지정한 사용자 ID에 해당하는 사용자의 모든 정보를 삭제하고 삭제 성공 여부를 반환합니다.	
Argument	*userID*	삭제할 사용자의 ID

■ **GetUserInfo**

Prototype	public IUserInfo GetUserInfo(string userID)	
Definition	지정한 사용자 ID에 해당하는 사용자의 사용자 정보를 가져옵니다.	
Argument	*userID*	사용자 정보를 가져올 사용자 ID

■ **CheckPassword**

Prototype	public bool CheckPassword(string userID, string password)

Definition	사용자의 패스워드가 맞는지 여부를 체크합니다.	
Argument	*userID*	패스워드를 체크할 사용자 ID
	password	체크할 패스워드

■ GetUserInfos

Prototype	`public IUserInfo[] GetUserInfos()`
Definition	모든 사용자의 정보를 가져옵니다.

■ DisableLogin

Prototype	`public void DisableLogin(string userID)`	
Definition	지정한 사용자 ID의 사용자를 로그인하지 못하도록 설정합니다.	
Argument	*userID*	로그인하지 못하도록 할 사용자 ID

■ EnableLogin

Prototype	`public void EnableLogin(string userID)`	
Definition	지정한 사용자 ID의 사용자를 로그인 가능하도록 설정합니다	
Argument	*userID*	로그인 가능하도록 할 사용자 ID

■ IsLoginEnabled

Prototype	`public bool IsLoginEnabled(string userID)`	
Definition	지정한 사용자 ID의 사용자가 로그인이 가능한지 여부를 가져옵니다.	
Argument	*userID*	로그인 가능 가능 여부를 체크할 사용자 ID

■ CreateGroup

Prototype	`public string CreateGroup(string groupName, string parentGroupID, string description)`	
Definition	새로운 그룹을 생성하고 생성된 그룹의 ID를 반환합니다..	
Argument	*groupName*	새로 생성할 그룹의 이름
	parentGroupID	새로 생성할 그룹의 상위 그룹 ID
	description	그룹에 대한 설명 내용

■ ModifyGroupName

Prototype	`public string ModifyGroupName(string groupID, string groupName)`
Definition	지정한 사용자 ID에 해당하는 사용자의 이름을 변경하고 변경 성공 여부를 반

Argument	환합니다.	
	groupID	그룹의 이름을 변경할 그룹 ID
	groupName	변경할 그룹의 이름

■ **ModifyGroupDescription**

Prototype	public bool ModifyGroupDescription(string groupID, string description)
Definition	지정한 사용자 ID에 해당하는 사용자의 설명을 변경하고 변경 성공 여부를 반환합니다.
Argument	*groupID* 그룹의 설명을 변경할 그룹 ID
	description 변경할 그룹의 설명

■ **DeleteGroup**

Prototype	public bool DeleteGroup(string groupID)
Definition	지정한 그룹 ID에 해당하는 그룹을 삭제하고 삭제 성공 여부를 반환합니다.
Argument	*groupID* 삭제할 그룹 ID

■ **CreateUserInGroup**

Prototype	public String createUserInGroup(String uName, String pwd, String gID, String desc) throws OZCPException
Definition	지정한 그룹에 새로운 사용자를 생성하고 생성된 사용자의 ID를 반환합니다..
Argument	*uName* 새로 생성할 사용자의 이름
	Pwd 패스워드
	gID 사용자를 생성할 그룹 ID
	desc 사용자에 대한 설명 내용

■ **TransferUser**

Prototype	public bool TransferUser(string userID, string newGroupID)
Definition	지정한 사용자의 그룹을 이동하고 그룹 이동 성공 여부를 가져옵니다
Argument	*userID* 그룹을 이동할 사용자 ID
	newGroupID 이동할 그룹 ID

■ **TransferGroup**

Prototype	public bool TransferGroup(string groupID, string targetGroupID)
Definition	지정한 그룹을 다른 그룹으로 이동하고 그룹 이동 성공 여부를 가져옵니다.

Argument	*groupID*	그룹을 이동할 그룹 ID
	targetGroupID	이동할 그룹 ID

■ **GetUserInfo**

Prototype	`public IUserInfo[] GetUserInfos(string groupID)`
Definition	지정한 그룹 ID에 등록되어 있는 모든 사용자의 정보를 가져옵니다.
Argument	*groupID* 사용자의 정보를 가져올 그룹 ID

■ **GetGroupInfo**

Prototype	`public IGroupInfo GetGroupInfo(string groupID)`
Definition	지정한 그룹 ID에 해당하는 그룹의 정보를 가져옵니다.
Argument	*groupID* 그룹 정보를 가져올 그룹 ID

■ **GetSubGroupInfos**

Prototype	`public IGroupInfo[] GetSubGroupInfos(string groupID)`
Definition	지정한 그룹의 하위 그룹 정보를 가져옵니다.
Argument	*groupID* 하위 그룹의 정보를 가져올 그룹 ID

■ **GetParentGroupInfo**

Prototype	`public IGroupInfo GetParentGroupInfo(string groupID)`
Definition	지정한 그룹의 상위 그룹의 정보를 가져옵니다.
Argument	*groupID* 상위 그룹의 정보를 가져올 그룹 ID

■ **GetGroupId**

Prototype	`public string GetGroupID(string userID)`
Definition	지정한 사용자가 속해 있는 그룹의 정보를 가져옵니다.
Argument	*userID* 그룹 정보를 가져올 사용자 ID

■ **AddGroupAdministrator**

Prototype	`public bool AddGroupAdministrator(string userID, string groupID)`	
Definition	지정한 그룹의 그룹 관리자를 추가하고 추가 성공 여부를 반환합니다.	
Argument	*userID*	그룹 관리자로 추가할 사용자 ID
	groupID	그룹 관리자를 추가할 그룹 ID

■ **RemoveGroupAdministrator**

Prototype	public bool RemoveGroupAdministrator(string userID, string groupID)	
Definition	지정한 그룹의 그룹 관리자를 해제하고 해제 성공 여부를 반환합니다.	
Argument	*userID*	그룹 관리자 권한을 해제할 사용자 ID
	groupID	그룹 관리자를 해제할 그룹 ID

■ **IsGroupAdministrator**

Prototype	public bool IsGroupAdministrator(string userID, string groupID)	
Definition	지정한 사용자 ID에 해당하는 사용자가 해당 그룹의 관리자인지 여부를 확인합니다.	
Argument	*userID*	관리자인지 체크할 사용자 ID
	groupID	그룹 ID

■ **GetGroupAdministrators**

Prototype	public IUserInfo[] GetGroupAdministrators(string groupID)	
Definition	지정한 그룹의 그룹 관리자 정보를 가져옵니다.	
Argument	*groupID*	그룹 관리자 정보를 가져올 그룹 ID

■ **Distribute**

Prototype	public Stream Distribute(string categoryID, bool isRecursive)	
Definition	지정한 카테고리의 아이템을 오즈 뷰어에서 인식하는 인증된 파일로 저장합니다.	
Argument	*categoryID*	카테고리 ID
	isRecursive	리커시브하게 압축할지 여부

■ **Upload**

Prototype	public bool Upload(string categoryID, Stream in)	
Definition	압축된 아이템을 지정한 카테고리에 압축 해제한 후 압축 해제 성공 여부를 반환합니다.	
Argument	*categoryID*	카테고리 ID
	in	압축한 스트림

■ **ModifyUserAuthorityToItem**

Prototype	public bool ModifyUserAuthorityToItem(string userID, string itemID, Authority permission)	
Definition	지정한 아이템에 대한 사용자의 권한 정보를 변경하고 변경 성공 여부를 가져옵니다.	
Argument	*userID*	사용자 ID
	itemID	아이템 ID
	permission	권한 정보

■ **ModifyUserAuthorityToCategory**

Prototype	public bool ModifyUserAuthorityToCategory(string userID, string categoryID, Authority permission)	
Definition	지정한 카테고리 ID에 대한 지정한 사용자 ID의 권한 정보를 변경하고 변경 성공 여부를 반환합니다.	
Argument	*userID*	사용자 ID
	categoryID	카테고리 ID
	permission	권한 정보

■ **ModifyGroupAuthorityToItem**

Prototype	public bool ModifyGroupAuthorityToItem(string groupID, string itemID, Authority permission)	
Definition	지정한 아이템에 대한 지정한 그룹 ID의 권한 정보를 변경하고 변경 성공 여부를 반환합니다.	
Argument	*groupID*	그룹 ID
	itemID	아이템 ID
	permission	권한 정보

■ **ModifyGroupAuthorityToCategory**

Prototype	public bool ModifyGroupAuthorityToCategory(string groupID, string categoryID, Authority permission)	
Definition	지정한 카테고리 ID에 대한 지정한 그룹 ID의 권한 정보를 변경하고 변경 성공 여부를 반환합니다.	
Argument	*groupID*	그룹 ID
	categoryID	카테고리 ID
	permission	권한 정보

■ **GetUserAuthorityToItem**

Prototype	`public Authority GetUserAuthorityToItem(string userID, string itemID)`	
Definition	지정한 아이템에 대한 지정한 사용자 ID의 권한 정보를 가져옵니다.	
Argument	*userID*	사용자 ID
	itemID	아이템 ID

■ **GetUserAuthorityToCategory**

Prototype	`public Authority GetUserAuthorityToCategory(string userID, string categoryID)`	
Definition	지정한 카테고리에 대한 지정한 사용자 ID의 권한 정보를 가져옵니다.	
Argument	*userID*	사용자 ID
	categoryID	카테고리 ID

■ **GetGroupAuthorityToItem**

Prototype	`public Authority GetGroupAuthorityToItem(string groupID, string itemID)`	
Definition	지정한 아이템에 대한 지정한 그룹 ID의 권한 정보를 가져옵니다.	
Argument	*groupID*	그룹 ID
	itemID	아이템 ID

■ **GetGroupAuthorityToCategory**

Prototype	`public Authority GetGroupAuthorityToCategory(string groupID, string categoryID)`	
Definition	지정한 카테고리 ID에 대한 지정한 그룹 ID의 권한 정보를 가져옵니다.	
Argument	*groupID*	그룹 ID
	categoryID	카테고리 ID

■ **GetGroupInfosOfItem**

Prototype	`public IGroupInfo[] GetGroupInfosOfItem(string itemID, Authority permission)`	
Definition	지정한 아이템 ID에 대해 permission 이상의 권한이 부여된 모든 그룹 정보를 가져옵니다.	
Argument	*itemID*	아이템 ID
	permission	권한 정보

■ **GetGroupInfosOfCategory**

Prototype	public IGroupInfo[] GetGroupInfosOfCategory(string categoryID, Authority permission)	
Definition	지정한 카테고리 ID에 대해 permission 이상의 권한이 부여된 모든 그룹 정보를 가져옵니다.	
Argument	*categoryID*	카테고리 ID
	permission	권한 정보

■ **GetUserInfosOfItem**

Prototype	public IUserInfo[] GetUserInfosOfItem(string itemID, Authority permission)	
Definition	지정한 아이템 ID에 대해 permission 이상의 권한이 부여된 모든 사용자 정보를 가져옵니다.	
Argument	*itemID*	아이템 ID
	permission	권한 정보

■ **GetUserInfosOfCategory**

Prototype	public IUserInfo[] GetUserInfosOfCategory(string categoryID, Authority permission)	
Definition	지정한 카테고리 ID에 대해 permission 이상의 권한이 부여된 모든 사용자 정보를 가져옵니다.	
Argument	*categoryID*	카테고리 ID
	permission	권한 정보

■ **GetCategoryInfosOfUser**

Prototype	public ICategoryInfo[] GetCategoryInfosOfUser(string userID, string categoryID, Authority permission)	
Definition	지정한 카테고리의 하위 카테고리 중에 지정한 사용자의 권한이 permission 이상인 모든 카테고리의 목록을 가져옵니다.	
Argument	*userID*	사용자 ID
	categoryID	카테고리 ID
	permission	권한 정보

■ **GetCategoryInfosOfGroup**

Prototype	public ICategoryInfo[] GetCategoryInfosOfGroup(string groupID, string categoryID, Authority permission)
Definition	지정한 카테고리의 하위 카테고리 중에 지정한 그룹 ID의 권한이 permission 이상인 모든 카테고리의 목록을 가져옵니다.

Argument	*groupID*	그룹 ID
	categoryID	카테고리 ID
	permission	권한 정보

■ **GetItemInfosOfUser**

Prototype	`public IItemInfo[] GetItemInfosOfUser(string userID, string categoryID, Authority permission)`
Definition	지정한 카테고리의 아이템 중에 지정한 사용자의 권한이 permission 이상인 모든 아이템 목록을 가져옵니다.

Argument	*userID*	사용자 ID
	categoryID	카테고리 ID
	permission	권한 정보

■ **GetItemInfosOfGroup**

Prototype	`public IItemInfo[] GetItemInfosOfGroup(string groupID, string categoryID, Authority permission)`
Definition	지정한 카테고리의 아이템 중에 지정한 그룹 ID의 권한이 permission 이상인 모든 아이템 목록을 가져옵니다.

Argument	*groupID*	그룹 ID
	categoryID	카테고리 ID
	permission	권한 정보

오즈 리파지토리 구현

리파지토리 매니저 호출 방식

리파지토리 매니저에서 API(oz.framework.cp.client.OZFrameworkAPI)를 사용하는 방법은 기존과 동일하며 리파지토리 매니저에서 API를 호출할 경우 아래와 같은 순서로 API가 처리됩니다.

- 리파지토리 매니저에서 서버에 요청하기 전에 리파지토리 매니저가 새로운 CP를 지원하는 버전인지 확인합니다.
- 리파지토리 매니저가 새로운 CP를 지원하는 버전일 경우 현재 호출한 매개 변수를 새 버전의 CP에 맞게 변환하여 새로 추가된 API를 호출합니다.
- 리파지토리 매니저가 새로운 CP를 지원하지 않는 버전일 경우 이전과 동일한 방식으로 API를 호출합니다.

※ 참고사항 : 새로 추가된 API는 서버의 RepositoryEx API와 형태가 동일하며 클라이언트에서 직접 사용하지 않습니다.

개발 시 주의 사항

- Adapter 추가 개발 시 IOZRepository 인터페이스를 반드시 구현하여야 합니다.
- Adapter 추가 개발 시 Category를 개발할 경우 반드시 Item을 개발하여야 하며 Group을 개발하는 경우에는 반드시 User를 개발하여야 합니다.
- 현재 리파지토리에서 지원하지 않는 기능은 다음과 같습니다.
  - 카테고리, 아이템 등을 삭제한 후 다시 복원하는 기능은 지원하지 않습니다.
  - 주석문 기능은 지원하지 않습니다.

Sample : oz.framework.api.RepositoryEx API Class

```
using System;
using System.IO;
using System.Reflection;
using System.Collections.Specialized;
using RepositoryEx = oz.framework.api.RepositoryEx;
using oz.framework.repositoryex.info;
```

```
namespace sample
{
                              /// <summary>
       /// 전체 경로를 ID로 사용하는 Repository에 접근하기 위한 API 샘플
                              /// </summary>
                     public class RepositoryExTest
                                   {
      public static void Main()
      {
         const string URL = "http://127.0.0.1/oz/server.aspx"; // 서버가 구동중인
URL
         const string ID = "admin"; // default
         const string PASSWORD = "admin"; // default

         using(RepositoryEx repository = new RepositoryEx(URL, ID, PASSWORD,
false))
         {
            // 테스트를 위해 임시로 리파지토리 설정을 변경합니다
            NameValueCollection prevConf = repository.GetConfiguration();

            NameValueCollection tempConf = new NameValueCollection();
            tempConf["REPOSITORY_TYPE"] = "BUILTIN";
            tempConf["REPOSITORY_FILE_PATH"] = Path.GetTempPath();
            tempConf["REPOSITORY_ITEM_NUMBER_PER_DIRECTORY"] = "100";
            tempConf["REPOSITORY_HISTORY_ITEM_VALID_DAYS"] = "20";
            repository.SetConfiguration(tempConf);
            repository.Reload();

            string itemID = null;

            string      categoryID      =      repository.CreateCategories(new
string[]{"test"}, new string[]{"/"}, "comment")[0];

            // 아이템 생성, 체크아웃/인, 변경, 삭제
            using(Stream                          @in                          =
Assembly.GetExecutingAssembly().GetManifestResourceStream("sample.parameter_tes
t.odi"))
            {
               // 아이템 생성
               itemID              =              repository.CreateItems(new
string[]{"parameter_test.odi"},   new   string[]{"Repository   API   test"},   new
string[]{"/test"},
                  new bool[1], new Stream[]{@in}, "comment")[0];

               if(null == itemID || 0 == itemID.Length)
               {
                  throw   new   Exception("Failed   to   create   item;   "   +
repository.GetLastErrorMessage(0));
               }
```

```
            }

            // 아이템에 대한 독점점 수정권한을 요청합니다
            Stream checkedOutItem = repository.CheckOut(new string[]{itemID},
new string[]{@"d:\"}, new long[1], new bool[1])[0];

            // 아이템의 수정 사항을 반영합니다
            repository.CheckIn(new   string[]{itemID},   new   bool[1],   new
Stream[]{checkedOutItem}, null, new bool[1]);

            // 아이템의 이름 수정
            itemID  =  repository.ModifyItemName(itemID,  "modified_item.odi",
"comment");

            string    tempCategoryID    =    repository.CreateCategories(new
string[]{"newCategory"}, new string[]{categoryID}, null)[0];

            Console.WriteLine(repository.GetItemInfo(itemID).ToString());

            repository.TransferItems(new string[]{itemID}, tempCategoryID);

            foreach(IItemInfo                  itemInfo                  in
repository.GetItemInfos(tempCategoryID))
            {
               Console.WriteLine(itemInfo.ToString());
               itemID = itemInfo.ID;
            }

            // 아이템 제거
            repository.DeleteItems(new string[]{itemID}, null, null);

            // 카테고리 제거
            repository.DeleteCategories(new string[]{categoryID}, new bool[1],
null);

            // 사용자 생성, 변경, 삭제
            string userID = repository.CreateUser("userName", "password", "/",
"description");

            userID = repository.ModifyUserName("userName", "brandNewName");

            repository.ModifyUserDescription(userID, "new description");

            repository.DeleteUser(userID);

            // 그룹 생성, 변경, 삭제
            string  groupID  =  repository.CreateGroup("newGroupName",  "/",
"description");

            userID = repository.CreateUser("newUserName", "password", groupID,
```

```
"description");

            groupID = repository.ModifyGroupName(groupID, "newGroupName2");

            repository.DeleteGroup(groupID);

            repository.SetConfiguration(prevConf);
            repository.Reload();
         }
      }
   }
}
```