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)**

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

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");
      }
   }
}
```

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);
```

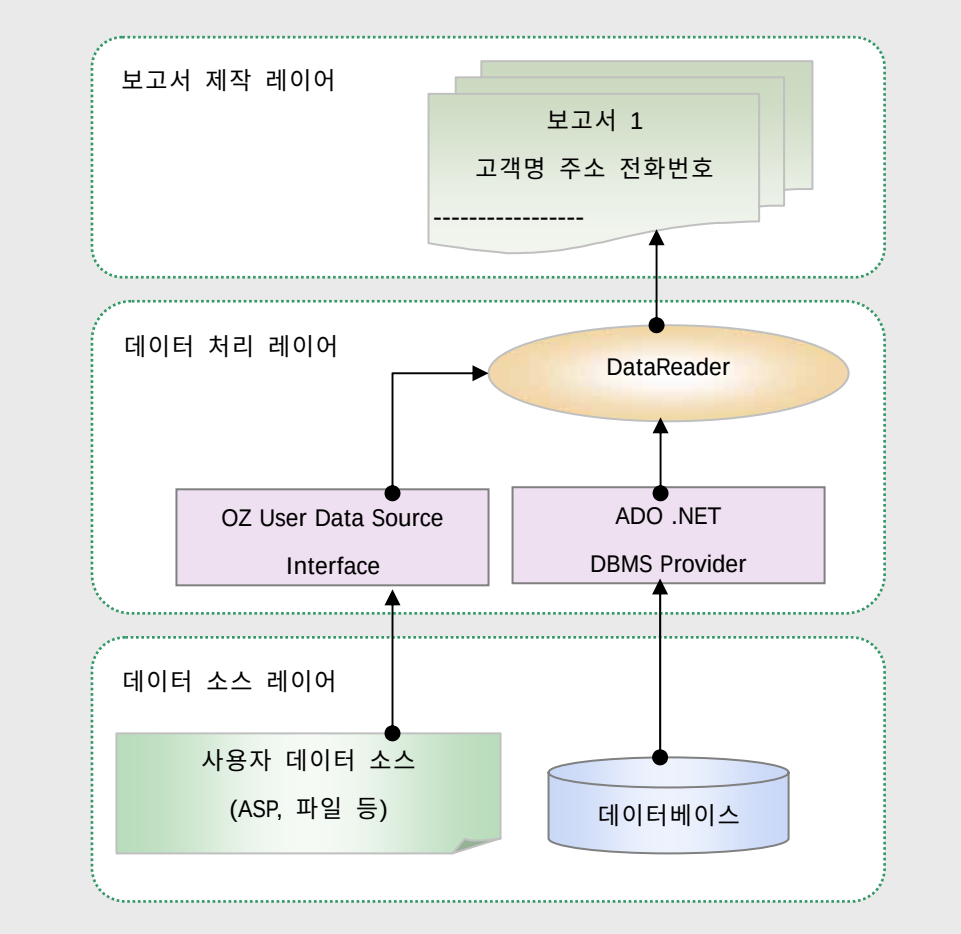
```
            s_repository.UpdateGroupID(prevGroupID, userID);
            s_repository.DeleteGroup(newGroupID);

            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. 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.

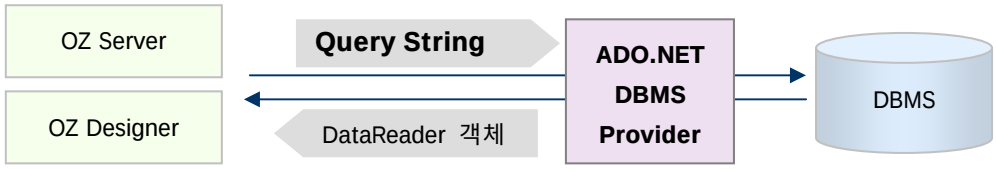


UDS 개발자는 다양한 형태의 사용자 데이터를 받아 들이는 방법과 이 데이터를 오즈가 인식할 수 있는 DataReader 객체로 만들어서 오즈로 넘겨주는 방법을 구현해야 합니다. 이렇게 구현된 프로그램은 파일로 만들어 보고서와 연동하게 됩니다. UDS 데이터셋은 일반 SQL 데이터셋과 동일하게 사용할 수 있습니다.

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

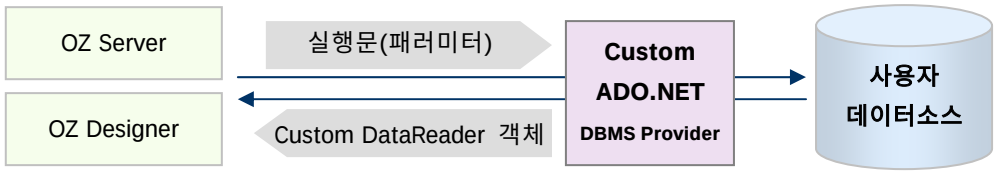
UDS for .NET 인터페이스

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.

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 데이터셋에 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.

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, IOZUDS_ConnectionPoolRef
   {
      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 System.Collections.Hashtable 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, IOZUDS_ConnectionPoolRef
   {
      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 System.Collections.Hashtable 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 = { {"data11","data12"}, {"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()
    {
    }
  }
}
```