

## 제 1 장 Caché DB 의 개요

### 1 Caché 란 무엇인가?

- 1) Caché 의 구조적 특성
- 2) 관계형 시스템과의 연계 및 적용이 용이한 고성능 오브젝트 데이터베이스

### 2 Caché 의 효과적인 보안 시스템

- 1) 인증: ID 설정
- 2) 허가: 사용자 접근권한 관리
- 3) 자원, 허가, 권한
- 4) 사용자와 역할
- 5) Audit: 사후 조사
- 6) 데이터베이스 암호화: 디스크 데이터의 보호

## 제 2 장 Caché 의 설치

### 1. Windows에 Caché 설치하기

- 1) 설치 필수 사항
- 2) Caché 업그레이드 설치
- 3) Caché 설치
- 4) 설치 후의 작업
- 5) 추가 고려 사항

### 2. Caché Studio 소개

- 1) Studio Window 개요
- 2) 프로젝트
- 3) 클래스 정의
- 4) CSP 파일
- 5) 루틴 편집기
- 6) 복수의 사용자 지원
- 7) Caché 문서 가져오기 / 내보내기
- 8) 디버깅
- 9) Caché 보안 시스템과의 통합
- 10) 소스 제어 흙(Hook)

### 3. Caché 터미널

- 1) Caché 터미널 시작하기
- 2) 특성
- 3) 사용 용도
- 4) ZWELCOME 루틴
- 5) 시작 네임스페이스
- 6) 터미널 프롬프트
- 7) 터미널에서 나오기

## 제 3 장 Caché ObjectScript

1. Caché ObjectScript 소개
  - 1) 특징
  - 2) 언어 개요
  - 3) 명령어, 함수의 호출
  - 4) ANSI 표준 M과의 관계
2. 변수
  - 1) 변수의 범주
  - 2) 변수 타입 정하기와 전환
  - 3) 변수 선언과 범위
3. 연산자
4. 명령어
  - 1) 명령어 인수
  - 2) 명령어의 후조건(postconditional) 표현식
  - 3) 코드 호출
  - 4) 지정 명령어
  - 5) 흐름 제어 명령어
  - 6) 입출력 명령어
  - 7) 기타 명령어
5. 트랜잭션 처리
  - 1) 응용프로그램에서의 트랜잭션 관리
  - 2) 자동 트랜잭션 롤백
  - 3) 시스템 레벨에서의 트랜잭션 이슈

## 제 4 장 Caché 오브젝트 모델

1. 오브젝트 참조- OREF, OID, ID
  - 1) OREF(Object Reference)
  - 2) OID(Object Identifier)
  - 3) ID(Identifier)
  - 4) OID 와 ID 값
  - 5) OREF 와 참조 카운트
2. 클래스 유형
  - 1) 비상주 (transient) 오브젝트 클래스
  - 2) 지속 (persistent) 오브젝트 클래스
  - 3) 시리얼(serial) 오브젝트 클래스
  - 4) 메모리 와 디스크에서의 임베디드 오브젝트
  - 5) 데이터 타입 클래스
3. 상속

- 1) 다중 상속
4. 클래스 컴파일

## 제 5 장 Caché SQL

1. Caché SQL 소개
  - 1) 아키텍처
  - 2) 기능
  - 3) 상호 운용성
2. Caché SQL 기초
  - 1) 테이블
  - 2) 쿼리
  - 3) 권한
  - 4) 조합(Collation)
  - 5) SQL 쉘
  - 6) SQL과 시스템 관리 포탈
3. 저장 프로시저의 정의와 사용법
  - 1) 개요
  - 2) 저장 프로시저 정의
  - 3) 저장 프로시저 사용 방법
4. Caché SQL 참조
  - 1) 집계(Aggregate) 함수
  - 2) 환경 설정
  - 3) 데이터 형식
  - 4) 날짜와 시간 구조 (Date and Time Constructs)
  - 5) 디폴트 사용자명과 패스워드
  - 6) 필드 제약
  - 7) 예약어
  - 8) 문자열 제어

## 제 6 장 다차원 배열, 글로벌의 이해

1. 특징
2. 글로벌의 구조
  - 1) 글로벌의 로직 구조
  - 2) 글로벌의 물리적 구조
  - 3) 글로벌 참조하기
3. 글로벌의 다차원 저장 방법
  - 1) 글로벌에서 데이터 저장
  - 2) 글로벌 노드 삭제하기

- 3) 글로벌 노드 존재 여부 테스트
- 4) 글로벌 노드 값 검색
- 5) 글로벌 노드 데이터 찾아가기
- 6) 글로벌 데이터 복사하기
- 7) 글로벌 내보내기
- 8) 글로벌 가져오기
- 9) 글로벌 공유 카운터 관리
- 10) 임시 글로벌 사용하기
- 11) 글로벌 데이터 정렬
- 12) 글로벌에 대한 간접 명령 사용
- 13) 트랜잭션 관리
- 14) 병행 처리 관리
- 15) 최근 참조 글로벌 확인

## 제 7 장 Caché Server Page 에 의한 Web 어플리케이션 구축

1. Caché Server Pages(CSP) 소개
  - 1) CSP 및 Zen
  - 2) 시작하기 전에
  - 3) 첫번째 CSP 페이지 생성
2. CSP 아키텍처
  - 1) CSP 구성 요소: 웹 서버, CSP Gateway, CSP 서버
  - 2) 웹 서버 URL 구성
  - 3) CSP Gateway 구성
  - 4) CSP 어플리케이션 옵션

## 제 8 장 Caché에서 SOAP 과 웹 서비스 사용하기

1. 소개
  - 1) Caché의 웹 서비스 지원
  - 2) Caché의 웹 클라이언트 지원
2. 웹 서비스 생성
  - 1) Caché 웹 서비스 개요
  - 2) 기본 필수 사항
  - 3) 카타로그 및 테스트 페이지
  - 4) 웹 서비스의 서비스명과 네임스페이스명
  - 5) 웹 서비스에서 지원되는 SOAP 버전
  - 6) WSDL 보기
3. 웹 메소드 생성
  - 1) 기본 필수 사항

- 2) 참조 또는 Output 인수로 값 리턴
  - 3) Input/Output 인수값으로 특수문자 사용
  - 4) 오브젝트를 Input/Output 인수로 사용
  - 5) 컬렉션을 Input/Output 인수로 사용
  - 6) 데이터 집합을 Input/Output 인수로 사용
  - 7) 클래스 쿼리를 웹 메소드로 사용하기
  - 8) SOAP 메시지에 대한 바인딩 스타일 명사하기
  - 9) SOAP 메시지 인코딩 설정
4. 웹 클라이언트 생성
    - 1) Caché SOAP 클라이언트 마법사 개요
    - 2) SOAP 클라이언트 마법사 사용하기

## 제 9 장 Zen 어플리케이션 개발

1. Zen 소개
  - 1) Zen 데모
  - 2) 지원 브라우저
  - 3) Zen 커뮤니티
  - 4) Zen의 장점
  - 5) 참고 문헌
2. Zen 컴포넌트 이용하여 개발하기
  - 1) Zen 테이블
  - 2) Zen report 만들기

## 제 10 장 Caché Jalapeño

1. 자바 Jalapeño Persistence 라이브러리
  - 1) 아키텍처
  - 2) 설치에 필요한 환경 및 설치
2. annotation 사용하기
  - 1) Annotation 기능
  - 2) 인덱스 추가
  - 3) 관계(Relationship) 추가
  - 4) 상위 클래스 추가
  - 5) 클래스와 프로퍼티 파라미터 추가
  - 6) 프로퍼티 타입 변경
  - 7) 스키마 생성 제어
3. Jalapeño 이클립스(Eclipse) 프로그인
  - 1) 프러그인 설정
  - 2) Jalapeño 메뉴 참조

## 제 11 장 Caché 오류 참조

1. 시스템 오류 메시지
  - 1.1 일반적인 시스템 오류 메시지
  - 1.2 새도잉 오류 메시지
  - 1.3 ANSI 표준 M 오류 메시지
2. SQL 오류 메시지
3. 개체 오류 메시지
4. CSP 오류

# 제 1 장 Caché DB 의 개요

## 1 Caché 란 무엇인가?

- 1) Caché 의 구조적 특성
- 2) 관계형 시스템과의 연계 및 적용이 용이한 고성능 오브젝트 데이터베이스

## 2 Caché 의 효과적인 보안 시스템

- 1) 인증: ID 설정
- 2) 허가: 사용자 접근권한 관리
- 3) 자원, 허가, 권한
- 4) 사용자와 역할
- 5) Audit: 사후 조사
- 6) 데이터베이스 암호화: 디스크 데이터의 보호

## 1. Caché 란 무엇인가?

Caché 는 미국 매사추세츠 캠브리지에 본사를 두고 있는 InterSystems 의 데이터베이스 관리시스템(DBMS) 제품이다. InterSystems 는 전세계 23 개국에 지사를 두고 있으며 의료 서비스 통합을 위한 최고의 플랫폼뿐만 아니라 다양한 산업분야에서 폭넓게 사용되고 있는 혁신적인 제품들을 제공하고 있다. TD Ameritrade, British Telecom, Cleveland Clinic, Deutsche Bank, Johns Hopkins Hospital, Kaiser Permanente, Merrill Lynch, Prudential Insurance Company, Siemens, Volvo 및 U.S. Army, VA Hospital 를 비롯하여, 그 외에도 수 많은 성공적인 기관들을 고객으로 가지고 있다.

Caché 는 극히 뛰어난 성능과 강력한 안정성을 가진 데이터베이스 시스템을 찾는 회사에게 적합하다. Caché 는 정교한 어플리케이션을 위해 신속한 개발 환경과 함께 강력한 오브젝트 데이터베이스를 견고한 SQL 과 독특하게 결합시킨 초고속 다차원 엔진을 제공한다. Caché 는 전세계 임상 의료 어플리케이션 중 가장 안정적인 최상의 데이터베이스이다.

Caché 의 주요 특징은 다음과 같다:

- 분산 데이터베이스 지원을 포함한 강력한 다차원 트랜잭션 엔진
- 객체와 고성능 SQL 을 결합시킨 통합 데이터 아키텍처
- 데이터베이스와 웹 어플리케이션을 빠르게 개발할 수 있는 관련 기술과 개발도구 제공
- 네이티브 객체 기반의 XML 과 웹 서비스 제공
- Java, EJB, JDBC, ActiveX, .NET, C++, ODBC, XML, SOAP, Perl, Python 등에 대한 상호 운용성 지원

이 장에서는 Caché 를 구성하는 주요 요소 및 기술에 대해 소개하며, 온라인을 통해 효과적인 기술지원 서비스를 이용하는 방법에 대해 설명한다.

### 1) Caché 의 구조적 특성

DBMS 를 통한 정보서비스 및 시스템 개발시, Caché 만이 보장할 수 있는 고성능 기술은 차별화된 구조에서 찾아볼 수 있다. 그 핵심은 Caché 데이터베이스 엔진을 통해 복잡한 데이터베이스 운영 시스템 구축에 필요한 데이터 저장, 병행 처리 관리, 트랜잭션, 프로세스 관리 등 전체적인 서비스를 매우 효과적으로 제공하는데 있다. Caché 의 엔진은 이러한 구조적 장점을 통해 강력한 데이터베이스 도구로서 다양한 요구가 반영된 성공적인 오브젝트 및 관계형 데이터베이스 관리시스템을 다음과 같이 구현하고 있다.

- 오브젝트 및 관계형 데이터베이스 시스템은 데이터베이스 엔진과 직접 연결되어 매우 효율적인 작업을 수행한다. 즉 오브젝트-관계형 미들웨어 또는 SQL-오브젝트 사이에 별도의 연결 기술이 필요하지 않다.

- 데이터베이스의 물리적 구성으로부터 논리적 구성의 분리는 기존 응용프로그램의 로직 변경 없이도 손쉽게 어플리케이션의 재배치를 가능하게 해 준다.
- 개방된 데이터베이스 엔진 인터페이스는 개발자가 필요할 때 언제든지 그 기능을 직접 활용하는 것을 가능케 해준다. 이러한 특성은 사용자 요구에 적합한 데이터베이스 관리 시스템 구축에서부터 성능 향상을 목표로 하는 핵심 어플리케이션 구축에 이르기까지 다양한 범위의 구축을 가능케 해 준다.
- Caché 아키텍처는 차세대 플랫폼으로서 기존 응용프로그램에 영향을 주지 않고 데이터베이스 엔진의 향상을 가능케 해준다. 예를 들면, 기존 응용프로그램 및 Caché 오브젝트 또는 관계형시스템에 어떠한 변경 없이도 확장성과 성능이 비약적으로 개선된 물리적 데이터 구조를 갖는 상위 Caché 버전으로의 마이그레이션이 가능하다. 또한 XML 과 같은 새로운 기술이 소개됨에 따라, Caché 는 이러한 기술을 자체적으로 지원하며, 기존 어플리케이션에 거의 영향을 주지 않고도 활용할 수 있는 고성능 컴포넌트들을 제공한다.

## 2) 관계형 시스템과의 연계 및 적용이 용이한 고성능 오브젝트 데이터베이스

Caché 는 현재 사용되고 있는 대부분의 관계형 데이터베이스 어플리케이션을 혁신적으로 업그레이드시키고 SQL 기반의 리포팅 툴을 지원할 뿐만 아니라, 관계형 모델의 한계를 극복할 수 있도록 설계되었다.

Caché 는 고성능 오브젝트 데이터베이스일 뿐만 아니라, 관계형 데이터베이스의 모든 특성과 기술을 또한 포함하고 있다. Caché 데이터베이스에 있는 모든 데이터는 관계형 테이블로 언제든 접근 가능하며 ODBC, JDBC 또는 오브젝트 메소드를 통한 표준 SQL 을 사용하여 조회하고 수정할 수 있다. Caché 데이터베이스 엔진의 강력한 성능으로 Caché 는 현재 관련 분야에서 사용되고 있는 제품중 가장 빠르고, 확장성이 뛰어나며, 신뢰성 높은 관계형 데이터베이스라고 할 수 있다. 또한, Caché 는 데이터의 표준화된 관계형 뷰를 지원하면서도 관계형 데이터베이스의 한계를 뛰어넘는 다음과 같은 장점들을 보여준다.

- 데이터를 오브젝트로 설계할 수 있는 특성(각 오브젝트는 생성시 자동적으로 관계형 표현으로 동기화된다)을 가지며 이러한 특성은 데이터베이스와 객체지향 어플리케이션 환경간의 임피던스 부정합 제거와 함께 관계형 데이터 모델링의 복잡성을 줄일 수 있다.
- 매우 간단한 오브젝트 기반의 병행처리 설계
- 사용자 정의 데이터 형식
- 데이터베이스 엔진내에서 다형성을 포함한 메소드와 상속을 이용할 수 있는 기능
- 오브젝트간의 식별과 관계를 다룰 수 있도록 SQL 을 위한 오브젝트 확장 기능
- 단일 응용프로그램내에서 SQL 과 오브젝트 기반 접근을 혼용하여 사용할 수 있는 기능
- 응용프로그램의 성능을 최대로 높이기 위해 데이터 저장에 사용되는 물리적 레이아웃, 클러스터에 대한 제어

오브젝트 및 관계형 접근이 모두 제공되는 대부분의 데이터베이스 제품들에서는 하나의 접근 방식이 다른 접근 방식의 기반위에서 제공되고 있는 반면, Caché 에서는 SQL 및 오브젝트 접근은 모두 데이터로 직접 연결되어 사용자들이 어떤 방식을 사용하더라도 성능이 향상된다.

## 2. Caché 의 효과적인 보안 시스템

Caché 의 고급 보안시스템은 간편하고 통합된 보안 아키텍쳐를 가지고 있으며 다음과 같은 특징을 가진다.

- 응용프로그램을 위한 강력하고 일관성 있는 고성능의 보안 구조를 갖추고 있다.
- 표준 인증시스템을 준수한다.
- 개발자가 어플리케이션에 보안 특성을 쉽게 적용할 수 있다.
- 시스템에 최소한의 부담만을 준다.
- Caché 는 보안의 강화가 필요한 시스템 환경에서도 효과적으로 운영되며, 다른 어플리케이션들과의 성공적인 운영을 보장한다.
- 보안정책 관리 및 강화를 위한 매우 유연한 기반구조를 제공한다.
- 효과적이고 실용적인 Caché 만의 고급 보안 시스템은 인증, 허가, Audit(감시), 데이터 암호화에 기반을 두고 있다.

### 1) 인증: ID 설정

인증이란 사용자가 누구인지를 어떻게 증명하는가를 말한다. 사용자가 다른 사람의 ID 를 도용해서 부정하게 얻어진 권한을 이용할 수 있기 때문에 믿을 만한 인증 절차 없는 서비스 사용에 대한 허가 메커니즘은 무의미하다고 할 수 있다.

인증 메커니즘은 Caché 를 어떻게 접근하느냐에 따라 다음과 같은 여러 방법을 사용할 수 있다.

- 커베로스(Kerberos) : 개방된 컴퓨터 네트워크 내에서 서비스 요구를 인증하기 위한 보안 시스템이다. 미국 MIT 대학의 Athena 프로젝트에서 출발하였고 가장 안정적인 방식으로 평가되고 있다. 사용자가 사이트를 이용할 때 ID 와 비밀번호를 한번 입력하게 되는데 실제로는 새로운 페이지를 열 때마다 인증을 필요로 하므로 처음 입력한 것을 암호화하고 물리적으로 안전한 장소에 보관하였다가 사용하는 방식을 쓰고 있다. 보안시스템이 미약한 네트워크에 대한 인증을 해주며, 다양한 사이버 공격으로부터 보호해준다. 또한, 암호는 네트워크상에서 절대 유출되는 일이 없도록 관리된다.
- 운영 체계 기반의 인증: 운영 체계 기반 인증은 Caché 사용자를 식별하기 위해 각 사용자들에 대한 운영체제 식별자를 사용한다. 윈도우즈에서 운영체제 기반의 인증은 각 로컬 사용자 로그인에만 사용된다. 윈도우즈 도메인 로그인에는 커베로스 인증이 사용된다.

- Caché 로그인: Caché 로그인은 사용자로부터 비밀번호를 입력받아 이전에 저장되었던 비밀번호와 비교한다. 시스템 관리자를 이용하여 비밀번호의 견고함을 보장하기 위해 비밀번호의 최소길이 등과 같은 비밀번호 기준을 구성할 수 있다.

Caché 는 인증 없이도 사용할 수 있는데, 안전하게 보호된 환경이나, 응용프로그램 또는 그 데이터가 공격의 대상이 되지 않을 경우에 적용한다.

## 2) 허가: 사용자 접근권한 관리

사용자가 일단 인증을 받으면, 다음 단계의 보안 관련 질문은 사용자가 어떤 항목에 대해 사용, 보기, 또는 변경할 수 있는 허가를 받았는지에 대한 내용일 것이다. 허가는 접근에 대한 결정과 제어이다. 허가는 사용자들과 정보자원간의 관계를 관리한다. 정보자원은 데이터베이스, Caché 서비스(CSP 접근 제어 등), 사용자가 생성한 응용프로그램등을 포함한다.

## 3) 자원, 허가, 권한

보안의 주요 목적은 여러 가지 정보와 이에 대한 사용 자격등 정보자원의 보호에 있다. Caché 에서 정보자원은 데이터베이스, 서비스, 응용프로그램, 도구는 물론, 관리 사항까지도 포함된다. 시스템 관리자는 허가를 해 줌으로써 이러한 정보자원에 대한 접근이 허용된다. 정보자원과 이와 함께 연관되어 주어진 허가를 권한이라 한다.

정보자원(resource)과 정보자산(asset )은 다음과 같은 점에서 다르다.

- 정보자산은 보호받을 항목이며, 정보자원은 Caché 보안 시스템에서 정보자산을 논리적으로 표현하는 것이다.
- 하나의 정보자원은 여러 정보자산을 보호할 수 있다.

## 4) 사용자와 역할

Caché 에서는 허가를 관리하는 방법으로 역할-기반 접근 관리(Role-Based Access Control, RBAC)를 사용한다. RBAC 에서 사용자는 정보자원을 다음과 같이 조작할 수 있게 된다.

- 정보자원은 허가에 연결되고 허가는 권한을 설정해 준다.
- 권한은 역할에 부여된다.
- 역할은 사용자 등의 멤버를 가진다.

사용자는 업무를 수행하기 위해 Caché 에 연결하는데, 이때 사용자는 자원에 접근하기 위한 일련의 권한을 가지고 있어야 한다. 이때 사용자에게 일일이 권한을 부여하는 것이 아니라, 역할을 통해서 권한을 가질 수 있도록 한다. 즉, 일련의 권한을 가지고 있는 역할을 먼저 생성한

다음에 사용자가 생성된 역할의 멤버가 되도록 함으로써 해당 역할이 가진 권한을 사용자가 부여 받는다. 이때 사용자는 하나 이상의 역할 멤버가 될 수 있다.

관리자는 역할이 가지고 있는 권한을 허가, 변경, 삭제할 수 있는데, 자동적으로 그 역할에 참여하는 다른 모든 사용자들에게 반영된다. 이를 통하여 관리자는 모든 사용자의 권한을 관리하는 대신 훨씬 적은 수의 역할만 관리하면 된다. 예를 들면, 병원의 응용프로그램은 진료를 담당하는 의사와 응급실 의사의 두 역할로 나누어, 각자의 역할에 적합한 권한을 부여 받을 수 있으며, 개별 사용자는 하나 또는 두 개 역할의 멤버가 된다.

Caché 는 역할에 대해 미리 정의를 내려 사용하며 사용자는 로그인 함으로서 사용자에게 미리 정의된 역할이 지정된다. 일단 로그인 되면 사용자는 Caché 응용프로그램이나, Caché 시스템 일부에서 일시적으로 추가된 역할의 멤버가 될 수 있다. 사용자에게 새로운 역할을 추가하는 것은 역할 확대(role escalation), 삭제하는 것은 역할 점감(role de-escalation)이라고 한다.

## 5) Audit: 사후 조사

Audit 를 통하여 시스템과 연관된 작동에 대해 검증할 수 있고 믿을 만한 자료를 얻을 수 있다. Audit 는 다음과 같은 여러 가지 보안 기능을 가지고 있다.

- 인증과 허가 절차상의 일어난 사항을 기록한다.
- 보안 관련 사고 발생시 발생 경과를 재구성할 수 있는 기반을 제공해준다.
- 이는 존재 자체만으로도 부정 사용에 대한 억제력으로 작용하여 공격자들의 부정 침입을 차단한다(공격자들은 공격 중에 자기 자신에 대한 정보를 노출할 것이라는 것을 알고 있음).

Audit 장치로 여러 시스템의 이벤트 뿐만 아니라 사용자 정의 이벤트도 기록 할 수 있다. 허가된 사용자들은 접근 관리 권한을 받으면 Caché 의 일부분인 도구를 사용, 이러한 Audit 기록을 근거로 보고서를 작성할 수 있다. Audit 기록이 민감한 정보를 가지고 있기 때문에 Audit 보고서를 작성하는 것 자체로 Audit 기록에 대한 개체를 만드는 것이다. 내장된 Caché 도구를 이용하여 Audit 기록 및 다른 태스크를 장기간 보관할 수도 있다.

## Caché Auditing 시스템

시스템 구성 환경에 따라 Audit 기록이 꽉 찼을 때 대처 방안이 달라질 수 있으며, Caché'에서는 이 대처방안을 변경할 수 있다. 즉, Audit 데이터베이스에 더 이상 쓰지 못하게 되었을 때 Caché 시스템을 중단 시키거나 가장 오래된 기록을 덮어쓰도록 할 수 있다.

## 6) 데이터베이스 암호화: 디스크 데이터의 보호

Caché 데이터베이스는 데이터를 보호하기 위해 암호화하여 저장 관리할 수 있다. 이는 사용자가 허가 받지 않은 데이터 (디스크에 저장된 데이터)를 보지 못하게 함으로써 데이터를 보호하게 된다. AES (Advanced Encryption Standard) 알고리즘을 사용, 기본적으로 128 비트 키를 통해 암호화를 구현한다. 암호화(encryption)와 복호화(decryption)는 Caché에서 디스크로 쓰거나 읽을 때 발생하는데, 취급되는 정보는 데이터 자체와 인덱스, 비트맵, 포인터, 할당 맵(allocation map), 충분 백업 맵 등이다.

데이터베이스 암호화 시스템에 경험 있는 사용자라면 암호화가 성능에 미치는 부작용을 걱정할 수도 있지만, Caché 플랫폼의 경우, 암호화와 복호화 과정은 최적화되어 있어 성능에 영향을 주지 않는다.

## 제 2 장 Caché 의 설치

### 1. Windows에 Caché 설치하기

- 1) 설치 필수 사항
- 2) Caché 업그레이드 설치
- 3) Caché 설치
- 4) 설치 후의 작업
- 5) 추가 고려 사항

### 2. Caché Studio 소개

- 1) Studio Window 개요
- 2) 프로젝트
- 3) 클래스 정의
- 4) CSP 파일
- 5) 루틴 편집기
- 6) 복수의 사용자 지원
- 7) Caché 문서 가져오기 / 내보내기
- 8) 디버깅
- 9) Caché 보안 시스템과의 통합
- 10) 소스 제어 흑(Hook)

### 3. Caché 터미널

- 1) Caché 터미널 시작하기
- 2) 특성
- 3) 사용 용도
- 4) ZWELCOME 루틴
- 5) 시작 네임스페이스
- 6) 터미널 프롬프트
- 7) 터미널에서 나오기

이 장에서는 Caché 설치과정을 버전 2009.1 을 기준으로 설명한다. 지원되는 플랫폼 및 브라우저 등은 Caché 온라인 문서 “Supported Platforms” 에서 확인할 수 있다.

#### 디폴트 Caché 설치 디렉터리

운영 체계	디렉터리
Microsoft Windows	C:\W\InterSystems\WCache (또는 CacheN : 둘 이상의 인스턴스가 설치되어 있는 경우)
Caché RPM kit	/usr/cachesys
UNIX-based	디폴트 위치 없음. 대상 위치를 지정해 주어야 한다
OpenVMS	디폴트 위치 없음. 대상 위치를 지정해 주어야 한다

이하 본 문서에서는 설치 디렉터리를 *install-dir* 로 명명한다.

### 1. Windows에 Caché 설치하기

사용자가 Windows 디렉터리 구조, 유틸리티, 명령어를 숙지하고 있는 것을 전제로 하며, 다음의 항목들에 대해 설명한다:

- 설치 필수 사항
- Caché 업그레이드 설치
- Caché 설치
- 설치 후의 작업
- 고려 사항

#### 1) 설치 필수 사항

Caché 설치를 위해서는 설치 유형에 관계없이 관리자 권한을 가져야 한다. 다음에서는 Caché의 신규설치 또는 업그레이드 설치에 필요한 소프트웨어, 하드웨어의 요건을 설명한다.

#### 디스크 공간

Caché 설치에는 CSP(Caché Server Pages) 지원을 포함하여 약 550 MB의 디스크 저장 공간(사용자 데이터에 필요한 디스크 공간은 별도)이 필요하다. 또한 설치 작업 자체를 위하여 추가적으로 10 MB의 디스크 공간이 요구된다.

설치 파일들에 대한 액세스를 위하여 설치 시스템에 장착되었거나 네트워크으로 공유된 DVD 드라이브를 사용할 수 있어야 한다. Windows가 원활하게 실행되는 시스템이라면 Caché를 실행하는데는 별다른 문제가 없으며 Caché 성능은 프로세서 성능 및 디스크 속도에 비례한다.

#### 지원되는 플랫폼과 웹 서버

현재 Caché 버전은 다음과 같은 Microsoft Windows 플랫폼을 지원한다:

- Windows 7 — 32-bit, 64-bit versions
- Windows Vista — 32-bit and 64-bit versions
- Windows Server 2008 — 32-bit, 64-bit, and Itanium versions
- Windows Server 2003 — 32-bit, 64-bit, and Itanium versions
- Windows XP Pro
- Windows 2000

**참조:** Caché 최신 버전 및 기술에 대한 정보는 온라인 문서 “Supported Platforms”에서 확인할 수 있다.

CSP 기술은 Microsoft IIS 웹 서버와 Apache 웹 서버 2.0, 2.2 버전을 지원된다. CSP를 사용하려면, Caché 설치 전에 웹 서버가 이미 설치되어 있어야 하며 이를 기반으로 Caché 설치 시 자동적으로 웹 서버에 대한 CSP 구성 설정하게 된다.

각 Caché 인스턴스에 있어서, 인스턴스 전용 Apache 웹 서버와 전용 CSP 게이트웨이가 설치되는데 이는 외부 환경(원도우상의 웹 서버 설치 여부)에 상관없이 CSP 어플리케이션인 시스템 관리 포탈과 Caché 온라인 문서를 실행하기 위함이다. Caché 인스턴스 웹 서버는 install-dir\httpd 디렉터리에 설치되며, 해당 Caché 인스턴스를 제거하면, 설치된 인스턴스 웹 서버도 제거된다.

## 업그레이드

이전 버전을 업그레이드하는 경우, 업그레이드 하기전에 Caché의 모든 작업을 종료하고 이전 버전의 데이터는 백업하여야 한다.

### 설치 디렉터리의 제한

디렉토리가 다음과 같은 경우 Caché 설치 디렉토리로 사용할 수 없다:

- 경로명에 탈자기호(^)가 있는 경우
- US ASCII 문자가 아닌 문자를 포함하고 있는 경우
- 루트 디렉토리(C:\ 등)
- \Program Files 의 하위 디렉터리인 경우

## 2) Caché 업그레이드 설치

설치 유형에 상관없이 Caché 업그레이드 절차는 동일하다. 업그레이드 설치는 기존 설치 유형에 기반한 구성요소들을 설치하는 작업으로 다음의 절차를 따른다:

1. 실행중인 Caché 서버는 중지시키며 또한 모든 Windows 응용프로그램을 종료한다.
2. DVD로부터 직접 설치하는 경우, DVD 삽입시 설치 프로그램이 자동 로드되지 않으며, Windows의 시작 -> 실행 메뉴를 선택하고 다음과 같이 입력한다:

**[drive]:\Wnt\Wsetup.exe**

여기에서, [drive]는 DVD 드라이브 문자이다. [확인] 버튼을 클릭하여 Caché 설치를 시작한다

3. 설치되어 있는 Caché 인스턴스(<인스턴스 디렉터리와 버전>) 목록이 나타난다.

**참고:** 설치된 인스턴스가 없는 경우, 바로 새로운 인스턴스를 설치하게 된다.

4. Caché 설치 대화 상자에는 다음과 같은 옵션이 표시되어, 사용자가 업그레이드를 관리할 수 있게 해준다.

- a. [선택적 사용자 정의 설치]를 선택하면, 업그레이드 설치 중에 원하는 구성 요소를 추가하거나 제거할 수 있다.

**참고:** 이전에 설치된 Caché 인스턴스를 업그레이드 할 때, 사용자가 변경하지 않으면 설치 프로그램이 모든 구성과 설정을 그대로 유지한다.

- b. [변경]을 선택하면, 사용자 정의 여부와는 관계없이, 다음 단계로 넘어간다.

5. Caché 설치 마법사가 설치의 성공적 완료를 표시하면 [마침]을 클릭한다.

### 3) Caché 설치

Caché 설치 절차는 기본적으로 동일하지만, 설치 유형에 따라 조금씩 차이가 있다.

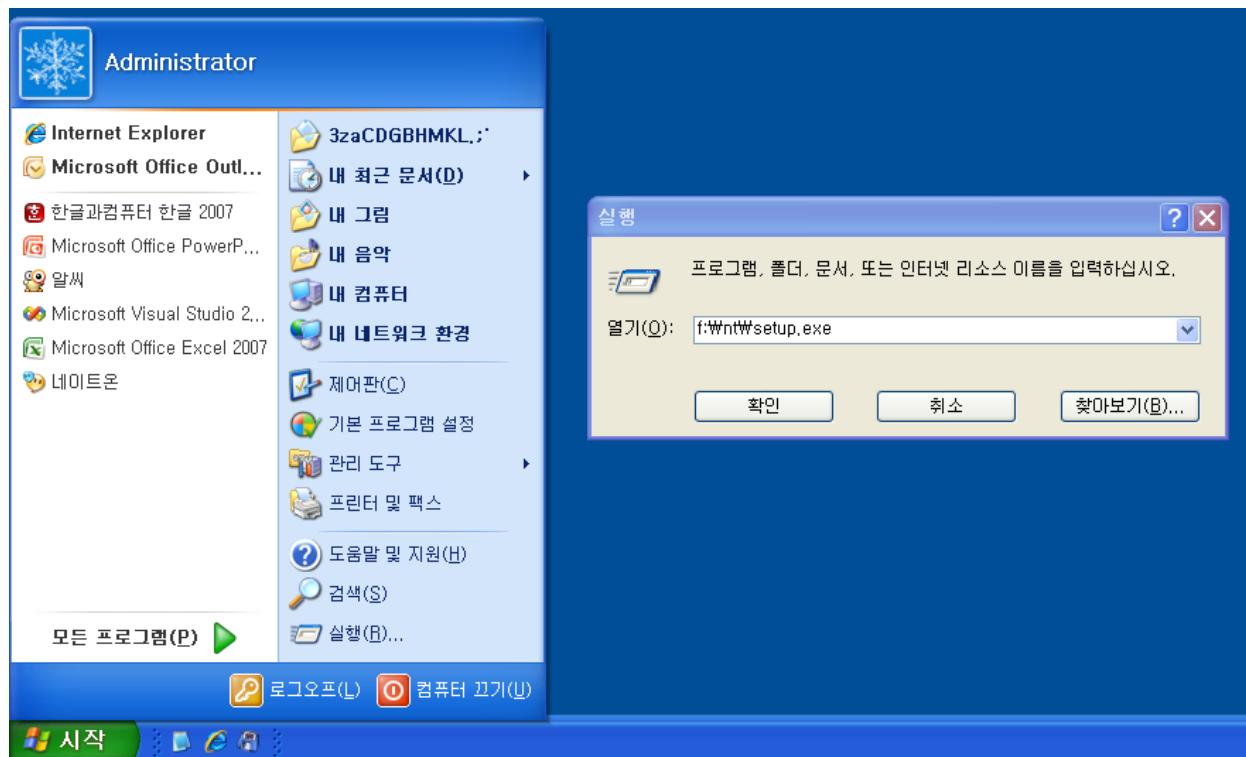
#### Caché 설치하기

선택한 설치 유형에 필요한 구성요소들을 다음의 과정에 따라 설치한다:

1. DVD에서 직접 설치하는 경우, DVD를 DVD 드라이브에 넣으면 설치 프로그램이 자동 로드된다.
2. 자동 로드되지 않는 경우, Windows 시작 -> 실행 메뉴를 선택하고 다음과 같이 입력한다:

**[drive]:\Wnt\Wsetup.exe**

여기에서 [drive]는 사용자의 DVD 드라이브 문자이다. [확인]를 클릭하여 Caché 설치를 시작한다.



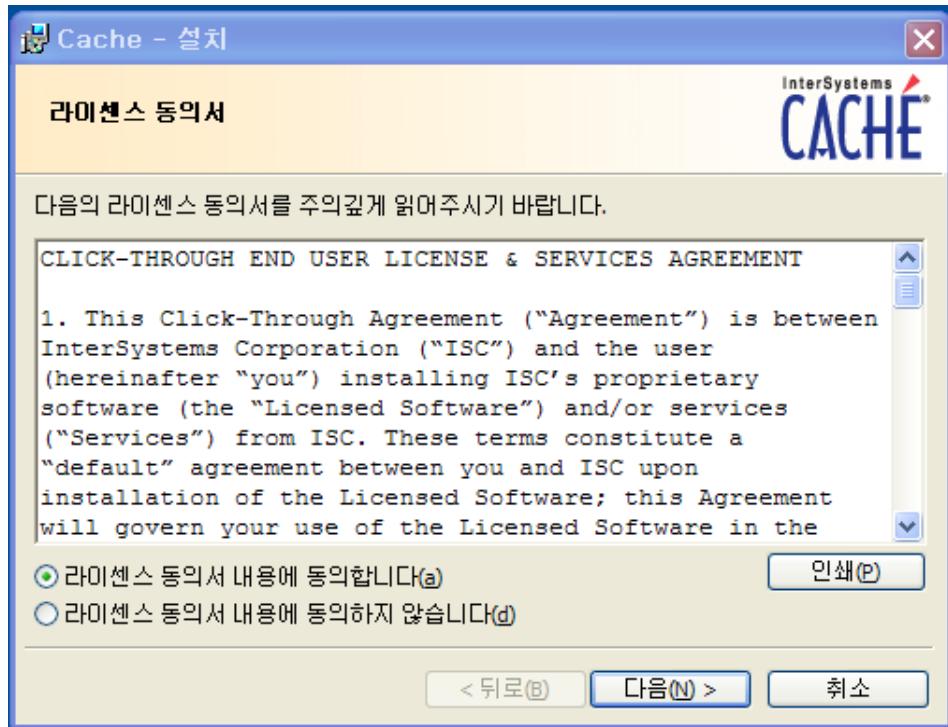
[그림 2-1] 실행명령을 이용한 Caché 설치



[그림 2-2] Caché 설치 초기화

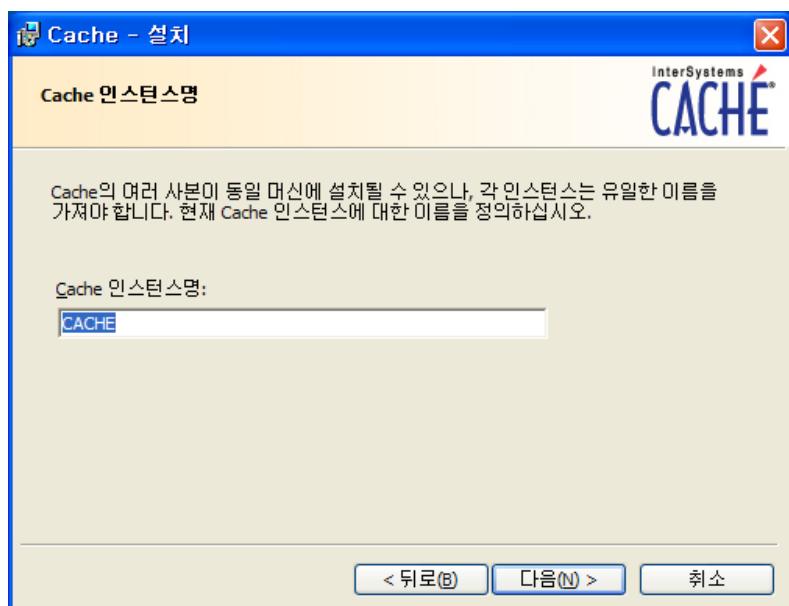
3. 이제 Caché 설치 프로그램이 시작되고, 다음의 버튼으로 설치 절차를 제어한다:
  - [다음] : 다음의 대화 상자로 이동
  - [뒤로] : 전 단계 대화 상자로 가서 작성한 것을 변경
  - [취소] : 설치 중지
4. 새로운 인스턴스를 설치하는 경우, 설치 프로그램은 라이센스 동의서(License Agreement)를 나타낸다. '라이센스 동의서 내용에 동의합니다'를 선택하여 라이센스 동의서에 동의함을 확인한다.

참고: 라이센스 동의서는 새 인스턴스를 설치하는 경우에만 나타나며, 라이센스 동의서에 동의하지 않으면 [취소] 버튼만 활성화 된다.



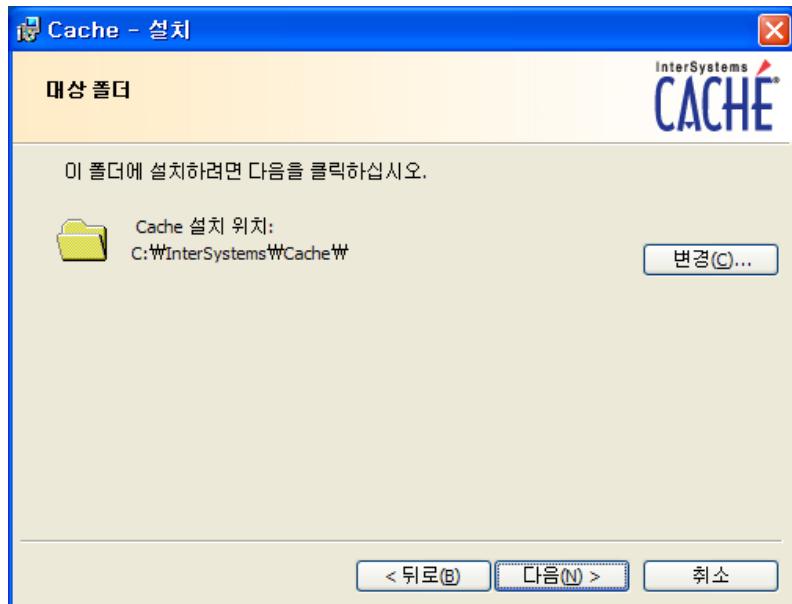
[그림 2-3] 라이선스 동의

5. Caché 인스턴스명 대화 상자에서, 설치하려는 새 인스턴스 이름을 입력한다. 디폴트 인스턴스명은 CACHE이며 다른 인스턴스가 설치되어 있는 경우에는 CACHEn 인데 n은 지금의 새로운 것을 포함한 Caché 인스턴스의 수를 의미한다. 기본을 선택하거나, 또는 원하는 이름을 입력한다.



[그림 2-4] Caché 인스턴스명

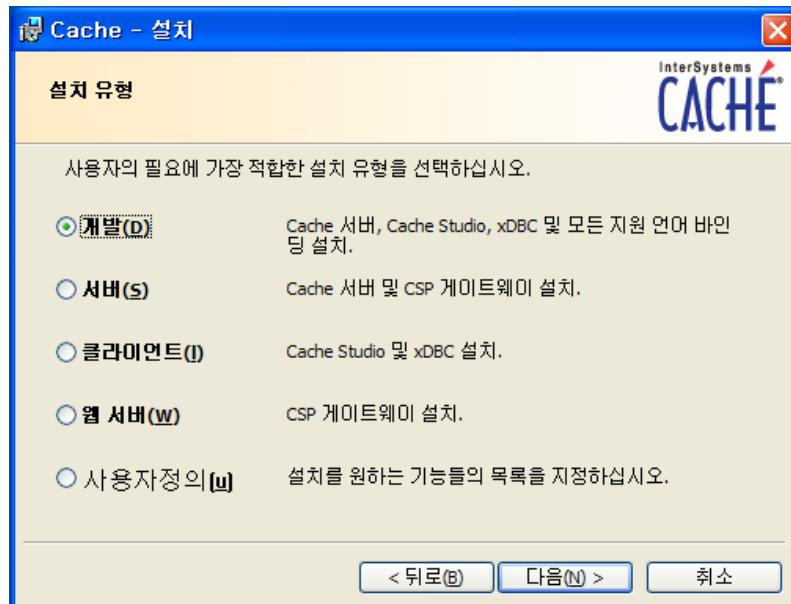
6. 대상 폴더 창에서는, 새 인스턴스에 대한 설치 디렉터리를 지정한다. 디폴트 위치는 C:\InterSystems\Cache (여러 개의 인스턴스가 있으면 CacheN)이다. [변경]을 클릭하여 원하는 디렉터리를 지정하거나 새로운 디렉토리를 설정한다. 지정한 디렉터리가 존재하지 않으면, 설치 프로그램이 새로 생성해 준다.



[그림 2-5] Caché 설치 대상폴더

7. 다음의 설치 유형 중 하나를 선택한다:

- **개발(D)** Caché 데이터베이스 엔진(Samples Database, User Database, SQL Gateway, Server Monitoring Tools), Caché Studio, 지원되는 모든 언어 바인딩, xDBC (ODBC, JDBC) 드라이버를 설치한다. 클라이언트/서버 업무를 모두 수행하려면 이 설치 유형을 선택한다.
- **서버(S)** Caché 데이터베이스 엔진(Samples database, User database, SQL Gateway, Server monitoring tools), CSP 게이트웨이를 설치한다. 이 유형은 Caché 데이터베이스 서버로 활용할 경우에 적합하다.
- **클라이언트(I)** Caché Studio와 xDBC(ODBC, JDBC) 드라이버를 설치한다. Caché 데이터베이스 서버의 클라이언트로 인스턴스를 활용하기 위해서는 이 설치 유형을 선택한다.
- **웹 서버(W)** CSP 게이트웨이(IIS, Apache 2.0, Apache 2.2)를 설치한다. CSP 게이트웨이에 필요한 컴포넌트들만 필요한 경우 이 설치 유형을 선택한다.
- **사용자 정의 설치(S)** 선택된 컴포넌트만을 설치 또는 삭제한다. 원하는 컴포넌트들만을 선택하여 설치 또는 삭제하고자 하는 경우, 이 설치 유형을 선택한다.



[그림 2-6] Caché 설치유형

다음은 설치 유형에 따라 어떤 구성 요소들이 설치되는지를 보여주는 테이블이다.

설치에 따른 구성 요소

설치유형	구성요소	개발	서버	클라이언트	웹서버
Caché 데이터베이스 엔진 (Caché 서버)	서버 모니터링 툴 Samples 데이터베이스 User 데이터베이스 SQL 게이트웨이	X	X		
Caché 실행 아이콘 (cube)		X	X	X	
Caché Studio		X		X	
xDBC	ODBC 드라이버 자바 데이터베이스 연결	X		X	
Caché 어플리케이션 개발	ActiveX 연결 C++ 바인딩 Light C++ 바인딩 자바 바인딩 C++ SDK Caché 엔진 링크 라이브러리 Perl 바인딩 Python 바인딩 .NET 바인딩 스레드 서버 라이브러리 기타 샘플		X		
문서	PDF 문서 온라인 문서	X	X		
웹 서버 게이트웨이 (CSP)	IIS용 CSP Apache 2.0.x용 CSP Apache 2.2.x용 CSP		X		X

[표 2-2] Caché 설치유형

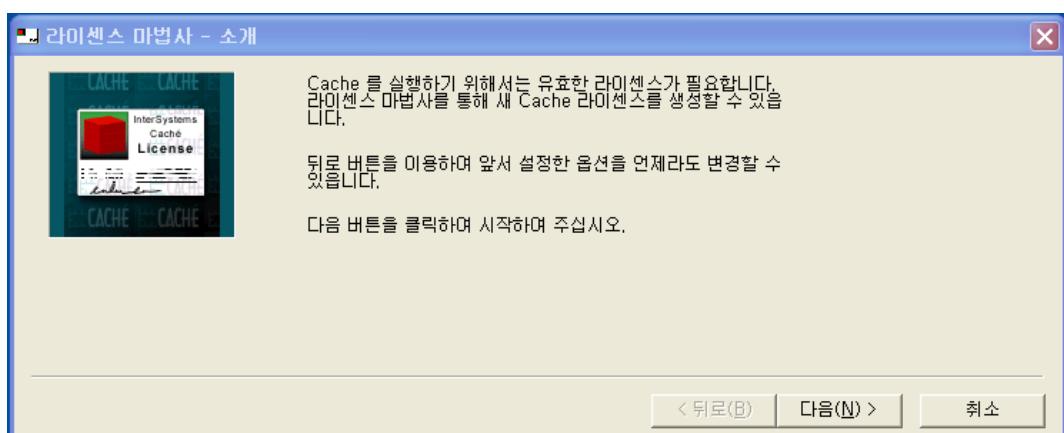
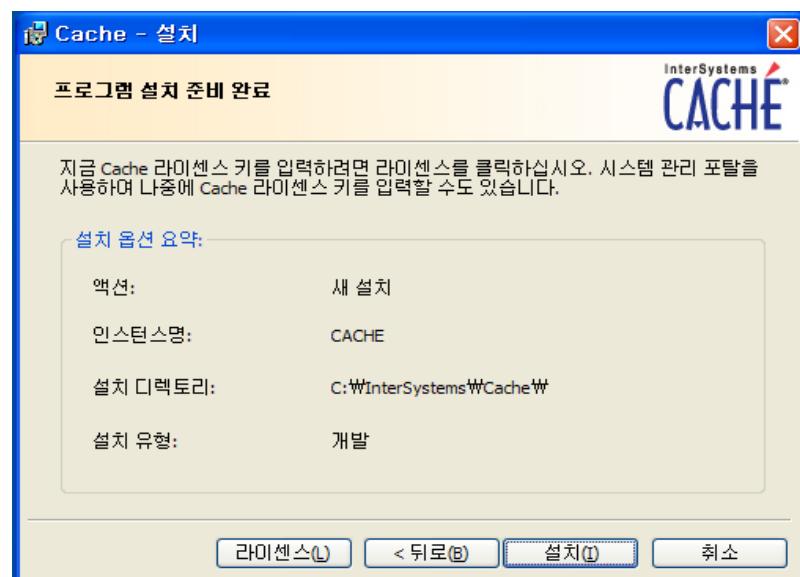
## Caché 개발 환경 설치

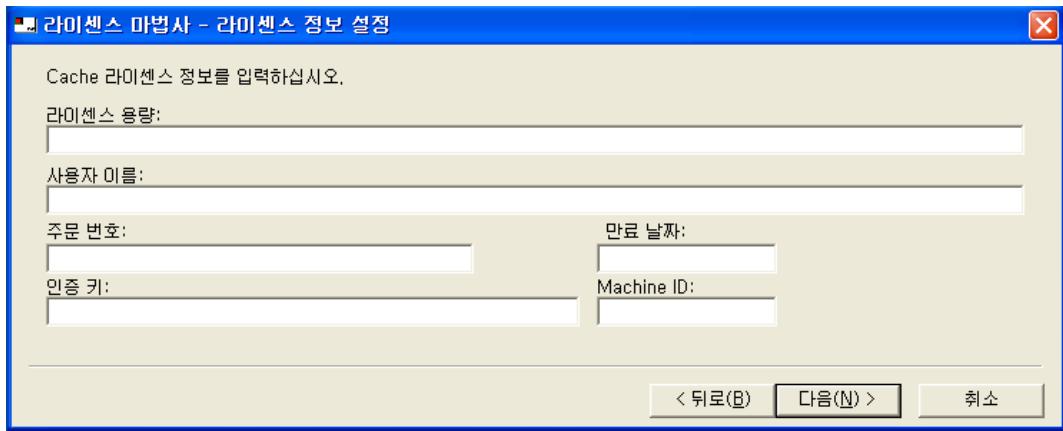
개발 환경에 필요한 Caché 구성요소들을 다음의 절차에 따라 설치한다:

1. Caché 설치를 시작한다.  
설치 유형 창에서 [개발] 버튼을 선택한다.
2. 설치 준비 창에서 정의한 설치명(인스턴스 명), 설치 유형, 및 설치 디렉토리를 검토한다.

**참고:** 라이센스 입력 창 – 설치 시스템에 Caché 라이센스 키가 설치되어 있지 않은 경우(즉 시스템 관리자 디렉터리, install\_dir\Wmgr, 에 Caché 키 파일이 없는 경우) 라이센스 입력창은 키가 없음을 표시하고 설치 중 라이센스 키를 입력할 것인지를 물어본다.

- 설치 중 라이센스 키를 입력하지 않고 계속 진행하려면, 선택 박스를 선택하지 않는다.  
라이센스 키는 설치가 완료된 후 시스템 관리 포탈의 **[홈] > [라이센스] > [라이센스 키]** 페이지를 통해 입력 가능하다.
- 설치 중 라이센스 키를 입력하는 경우, 선택 박스를 선택하여 Caché 라이센스 마법사를 실행하고 요구하는 라이센스 정보를 입력한다.





[그림 2-7] 라이센스 키 등록

- ‘설치(I)’ 버튼을 클릭하여 설치를 진행한다. 설치가 성공적으로 완료되면 *Getting Started* 페이지를 볼 것인지 여부를 선택하고 [마침]을 클릭한다.

Caché가 정상적으로 설치되면, Caché 큐브 아이콘이 Windows 작업 표시줄의 알림 영역에 나타난다. 큐브 아이콘을 클릭하면 Caché 운영에 필요한 메뉴가 나타난다. 또한 Windows 프로그램 메뉴에 Caché 항목이 표시된다. Windows 서버의 경우, 설치중 정의된 Caché 인스턴스명을 사용하여, Caché Windows 서비스명을 설정한다. Caché 서비스는 Windows 가 시작할 때 윈도우 서비스로서 자동 설정된다.

Caché 시스템 관리 포탈의 [총] > [환경 구성] > [메모리 및 시작] 페이지에서 시스템 구동 시 자동 시작 옵션을 변경할 수 있다. 만약 현재 인스턴스가 Windows 클러스터의 멤버라면, 자동 시작 옵션을 선택하면 안되며, 클러스터 관리자가 Caché를 시작하도록 하여야 한다.

### Caché 서버 시스템 설치

Caché 를 서버 시스템으로 사용하고자 하는 경우, 해당 구성요소들을 다음의 절차에 따라 설치한다:

1. Caché 설치를 시작한다

설치 유형 창에서 [서버] 버튼을 선택한다.

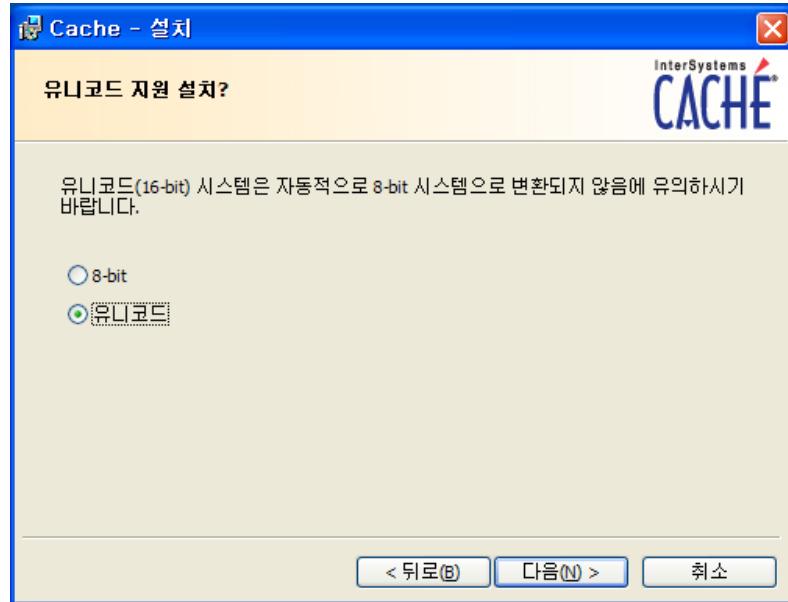
2. 유니코드 지원 설치 여부를 묻는 창에서 8 비트 또는 유니코드중 설치를 원하는 코드 체계를 선택할 수 있다 (디폴트는 윈도우 로케일에 따라 설정된다).

- **8 비트**: 소프트웨어는 문자를 8 비트 형식으로 취급한다.
- **유니코드**: 소프트웨어는 문자를 유니코드 형식으로 취급한다. 사용자의 응용프로그램이 한국어나 일본어처럼 유니코드 형식으로 데이터를 저장하는 경우, 유니코드를 선택한다.

InterSystems 에서는 Latin-1 문자 집합, ISO 8859-1 에 기반한 로케일인 경우 8 비트 문자 지원을 권장한다. 로케일의 기본 문자 설정이 Latin-1 이 아니거나, 다른 문자 집합에 근거한 로케일로부터 데이터를 액세스하려면, 유니코드를 사용하여야 한다. Caché의 8 비트 버전의 데이터인 경우, 다른

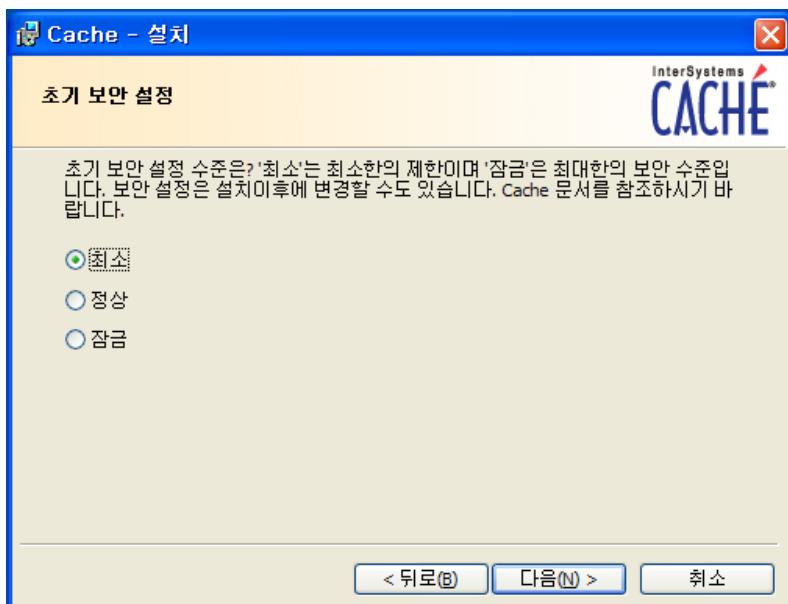
문자 집합에 기반한 8 비트 로케일로 옮길 수 없다.

**주의:** 유니코드 버전으로 설치한 후, 8 비트 버전으로 업데이트하는 경우, 데이터 손실 가능성이 있다. 이는 Caché 의 8 비트 버전에서는 16 비트(유니코드) 문자 데이터를 이해할 수 없기 때문이다.



[그림 2-8] 유니코드 지원 설치

- 초기 보안 설정 창에서는, 초기 Caché의 보안 수준을 어느 정도로 할 것인지를 결정한다. 최소를 선택하는 경우, 다음 설치 단계가 진행된다.



[그림 2-9] 초기 보안 설정

**참고:** 초기 보안 설정으로 ‘최소’를 선택했을 때, Caché에서 네트워크상의 공유 드라이브와 프린터를 사용하는 경우, 서버 시스템상의 로컬 관리자 권한을 갖는 계정을 선택하거나 새로 생성하여 Caché 서비스를 실행할 수 있는 계정으로 Windows 사용자 계정을 반드시 변경하여야 한다.

‘보통’이나 ‘장금’을 선택하면, Caché 서비스를 아래의 계정으로 실행할 수 있는 계정정보 입력창이 나타난다:

- 디폴트 시스템 계정: Caché를 실행할 수 있는 Windows 로컬 시스템 계정.
- 정의된 윈도우 사용자 계정: 설치 작업은 %All 역할(Role)을 가진 Caché 계정을 생성하여, Caché 관리에 필요한 서비스를 액세스할 수 있는 권한을 부여한다. 사용자가 이 계정의 패스워드를 입력하여 확인하여야 하며, 패스워드는 보안 설정에 맞는 기준을 만족해야 한다.

**참고:** 커베로스(Kerberos)를 사용하는 경우, 반드시 Caché 서비스를 실행하도록 정의된 사용자 계정을 입력해야 한다. InterSystems에서는 사용자가 이 목적으로 특별히 정의된 별도의 계정을 사용할 것을 권장한다.

‘다음’을 클릭하면, 규정된 사용자 계정을 입력하였는지를 다음과 같이 검증한다:

- 입력된 계정이 도메인에 존재하는지.
- 정확한 패스워드를 입력했는지.
- 입력된 계정이은 서버 시스템의 로컬 관리자 권한을 가지고 있는지.

**주의:** 설치 시스템이 Windows Vista 인 경우, 계정을 입력하기 전에 로컬 관리자 권한을 가지고 있는지 미리 확인하여야 한다. 왜냐하면, Vista 플랫폼의 경우, 설치 프로그램에서 이것을 확인할 수 없기 때문이다.

#### 4. 설치 준비 창에서는 제공된 설치명, 설치 유형, 설치 디렉터리를 다시 한 번 점검한다.

**참고:** 라이센스 입력 창 – 설치 시스템에 Caché 라이센스 키가 설치되어 있지 않은 경우(즉 시스템 관리자 디렉터리, install\_dir\Wmgr, 에 Caché 키 파일이 없는 경우) 라이센스 입력창은 키가 없음을 표시하고 설치 중 라이센스 키를 입력할 것인지를 물어본다.

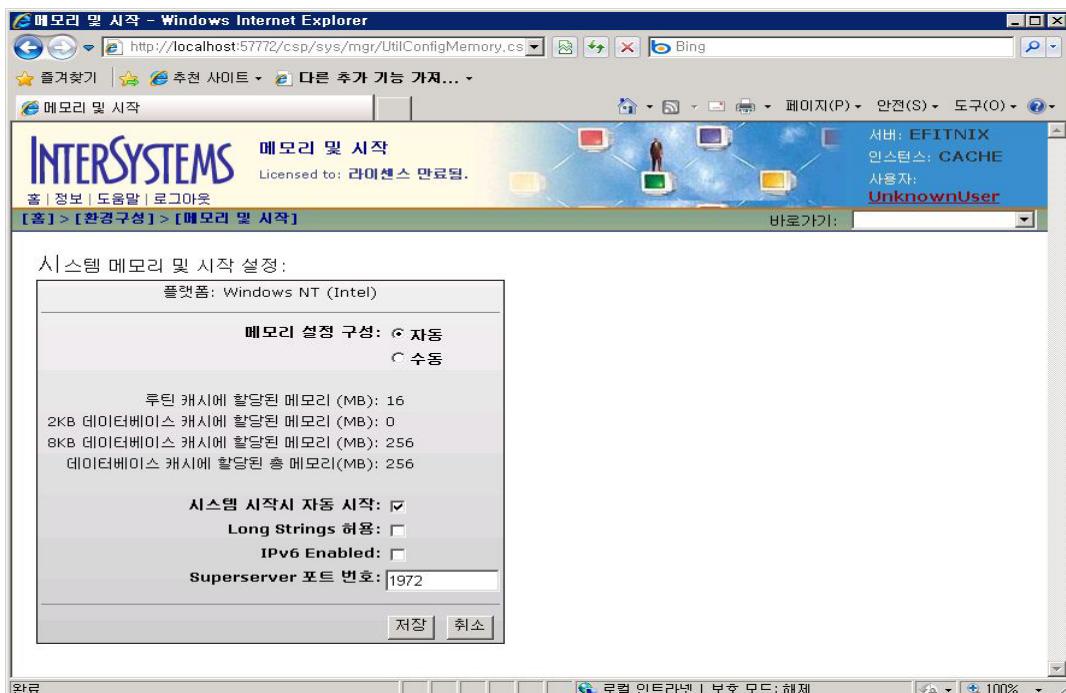
- 설치 중 라이센스 키를 입력하지 않고 계속 진행하려면, 선택 박스를 선택하지 않는다. 라이센스 키는 설치가 완료된 후 시스템 관리 포탈의 [홈] > [라이센스] > [라이센스 키] 페이지를 통해 입력 가능하다.
- 설치 중 라이센스 키를 입력하는 경우, 선택 박스를 선택하여 Caché 라이센스 마법사를 실행하고 요구하는 라이센스 정보를 입력한다.

'설치'를 클릭하면 지정된 설치 디렉터리에 Caché가 설치된다.

5. 설치가 성공적으로 완료되면 사용자는 시작 페이지를 볼 것인지를 선택하고 [마침]을 클릭한다.

Caché가 정상적으로 설치되면, Caché 큐브 아이콘이 Windows 작업 표시줄의 알림 영역에 나타난다. 큐브 아이콘을 클릭하면 Caché 운영에 필요한 메뉴가 나타난다. 또한 Windows 프로그램 메뉴에 Caché 항목이 표시된다. Windows 서버의 경우, 설치중 정의된 Caché 인스턴스명을 사용하여, Caché Windows 서비스명을 설정한다. Caché 서비스는 Windows 가 시작할 때 윈도우 서비스로서 자동 설정된다.

Caché 시스템 관리 포탈의 [환경 구성] > [메모리 및 시작] 페이지에서 시스템 구동 시 자동 시작 옵션을 변경할 수 있다. 만약 현재 인스턴스가 Windows 클러스터의 멤버라면, 자동 시작 옵션을 선택하면 안되며, 클러스터 관리자가 Caché를 시작하도록 하여야 한다.



[그림 2-10] 시스템 시작시 자동 시작 설정

### Caché 클라이언트 설치

Caché 클라이언트에 필요한 구성요소들을 다음의 절차에 따라 설치한다:

1. Caché 설치를 시작한다.

설치 유형 창에서 [클라이언트]를 선택한다.

2. 설치 준비 창에서 설치명(인스턴스명), 설치 유형, 설치 디렉터리를 선택한다.

[설치]를 클릭하면 지정된 설치 디렉터리에 Caché가 설치된다.

3. 설치마법사 완료 창은 설치가 성공적으로 완료되었음을 나타낸다. [마침]을 클릭한다.

Caché 클라이언트가 설치되면 Caché 큐브 아이콘(■)이 Windows 작업 줄의 알림 영역에 흐리게 나타난다. 이는 현재 클라이언트를 설치한 로컬 시스템에 Caché 서버가 없거나 실행중이 아니기 때문이다. 클라이언트를 사용하기 전에, 먼저 대상 서버를 선택한다.

## 웹 서버 설치

CSP 게이트웨이에 필요한 Caché 구성요소들을 다음 절차에 따라 설치한다:

1. Caché 설치를 시작한다.  
설치 유형 창에서 [웹 서버]를 선택한다.
2. 설치 준비 창에서 설치명(인스턴스명), 설치 유형, 설치 디렉터리를 선택한다.  
[설치]를 클릭하면 지정된 설치 디렉터리에 Caché가 설치된다.
3. 설치마법사 완료 창은 설치가 성공적으로 완료되었음을 나타낸다. [마침]을 클릭한다.

웹 서버 게이트웨이(CSP 게이트웨이)를 설치 중에, 웹 서버가 실행 중이면 웹 서버의 중지를 요구하는 창이 나타난다. [예]를 선택하면, 설치 프로그램은 웹 서버를 중단시키고 CSP 게이트웨이를 설치한 다음, 중단된 웹 서버를 재실행한다. [아니오]를 선택하면, 웹 서버에 대한 CSP 게이트웨이 환경을 구성하지 않고 다만 Caché에 포함된 전용 Apache 웹 서버를 위한 전용(private) CSP 게이트웨이만을 설치하게 된다.

설치 프로그램은 두 가지 유형의 웹 서버를 감지하고 CSP 게이트웨이 구성은 자동적으로 설정하는 옵션들을 제공한다. 다음 웹 서버중 감지된 서버는 체크되어 표시된다:

- CSP IIS
- CSP for Apache 2.x

선택한 옵션에 따라, Caché는 CSP 게이트웨이가 설치하고 해당 웹 서버에 CSP 게이트웨이 환경을 구성한다. CSP 게이트웨이 컴포넌트는 선택하고 웹 서버는 선택하지 않는 경우, CSP 게이트웨이에 필요한 모든 파일들은 설치되지만, CSP 게이트웨이를 위한 웹 서버 환경은 구성되지 않는다. 이 경우, 설치가 완료된 후에 수동적으로 CSP 게이트웨이 응용프로그램에 대한 구성을 웹 서버 환경에 구성할 수 있다.

모든 옵션을 선택하는 경우, CSP 게이트웨이가 설치되고 각 웹 서버 환경이 구성된다. CSP 게이트웨이는 현재 인스턴스에 대해 구성된 서버를 가리키는(?) 다음과 같은 CSP 어플리케이션 경로를 구성한다:

- /
- /csp
- /Cache (instance name)

**주의:** CSP 게이트웨이 설치는 이전 설치된 모든 CSP 게이트웨이 구성 및 웹 서버 구성은 대체한다. 특히 CSP 어플리케이션 경로 / 과 /csp 에 대한 이전의 설정을 변경된다.

Caché 는 OS 웹 서버와는 독립적으로 시스템 관리 포털과 온라인 문서를 서비스하기 위해 설정된 웹 서버 포트를 통해 실행되는 임베디드 Apache 웹 서버 및 전용(private) CSP 게이트웨이를 서버 시스템내에 설치한다.

### Caché 사용자 정의 설치

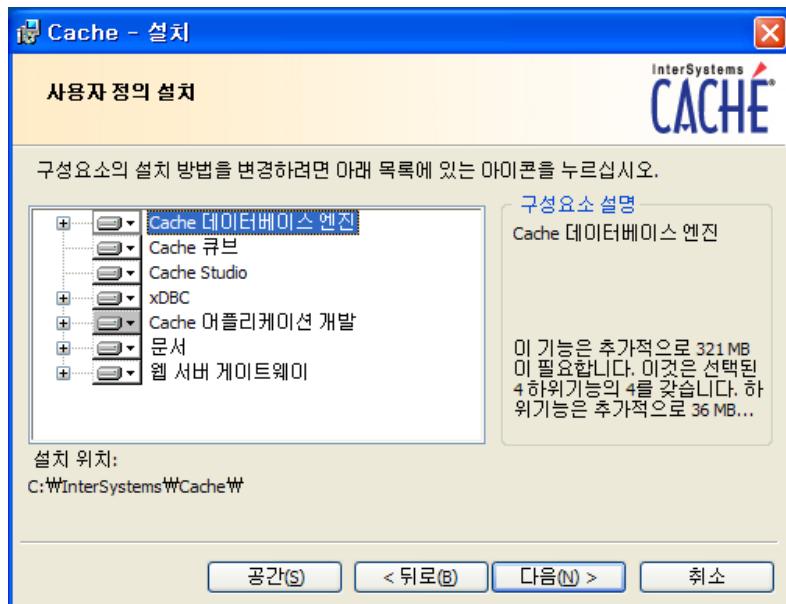
Caché 설치 프로그램에서는 사용자가 컴퓨터에 설치할 Caché 구성 요소를 선택할 수 있다. 예를 들면, 웹 서버(CSP) 게이트웨이만을 설치할 수도 있다. 그러나 선택에 따라 다른 구성요소를 함께 설치해야 하는 경우도 있다.

Caché 사용자 정의 설치 절차는 다음과 같다.

1. Caché 설치를 시작한다.

설치 유형 창에서 [사용자 정의 설치]를 선택한다.

2. 사용자 정의 설치 대화 상자에서, 설치에 따른 구성 요소 표에 설명된 것처럼 설치하려는 구성 요소를 선택한다.



[그림 2-11] 사용자 정의 설치

**중요:** Caché 데이터베이스 엔진(Caché 서버) 컴포넌트 전체 또는 일부를 선택적으로 설치하려면, (Caché 어플리케이션 개발 컴포넌트에 포함된) ActiveX 연결이 반드시 필요하다. 또한 문서 컴포넌트 전체 또는 일부를 설치하려면, Caché 데이터베이스 엔진(Caché 서버) 컴포넌트 전체가 필수적이다.

**참조:** 이전 설치된 구성 요소는, X 메뉴 항목을 선택하여 제거할 수 있다.

3. [공간] 버튼을 클릭하여, 선택된 구성 요소를 위한 충분한 공간이 디스크에 있는지를 확인 할 수

있다.

4. 유니코드 지원 설치 창에서는 [8 비트 또는 유니코드 지원]을 선택한다(디폴트 설정은 운영 체계로 케일에 따라 결정된다).

- 8 비트: 8 비트 형식의 문자집합을 이용한다
- 유니코드: 유니코드 형식의 문자집합을 이용한다. 사용자의 응용프로그램이 한국어처럼 유니코드 형식으로 데이터를 저장하는 경우, 유니코드를 선택한다.

InterSystems에서는 Latin-1 문자 집합 ISO 8859-1에 기반한 로케일에 대해서는 8 비트 문자 지원을 권장한다. 로케일의 기본 문자 설정이 Latin-1이 아니거나, 다른 문자 설정에 근거한 로케일로부터 데이터를 사용하려면, 유니코드를 사용한다. Caché의 8 비트 버전을 사용한다면, 데이터는 다른 문자 설정에 근거한 8 비트 로케일로 옮길 수 없다.

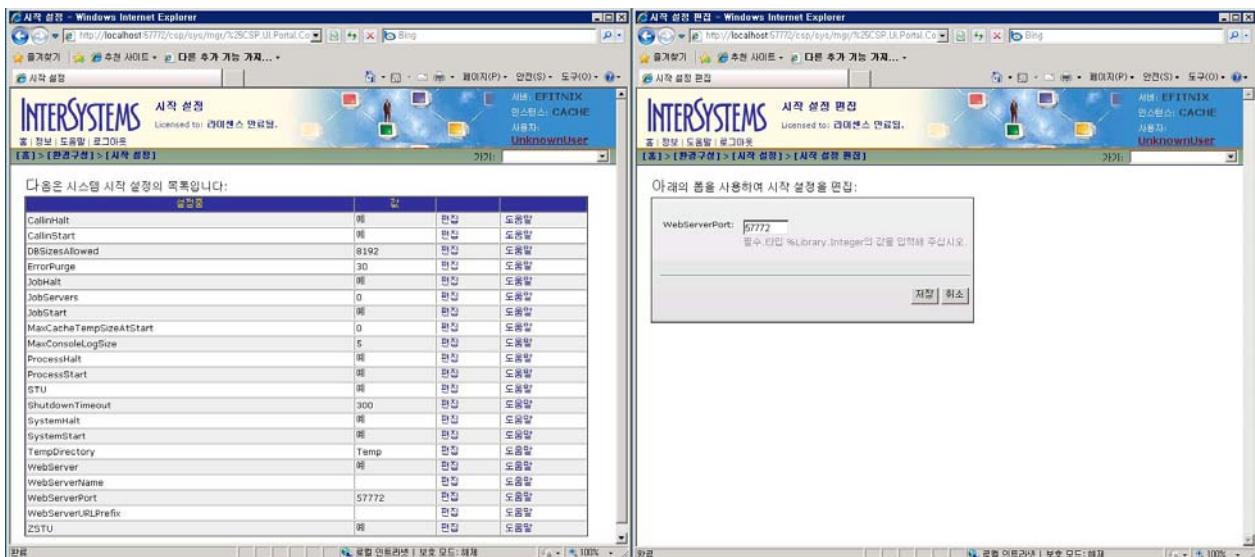
**주의:** 유니코드 설치 후 8 비트 버전으로 되돌아 오는 경우, 데이터 손실 가능성이 있기 때문에 지원되지 않는다. 이는 Caché의 8 비트 버전의 데이터베이스에서 16 비트 문자 데이터를 찾을 수 없기 때문이다.

5. 포트 번호 입력 창에서 Caché 서버와의 통신을 위한 전용 포트 번호를 설정한다.

**참고:** 65535 이상이거나 1 이하의 포트 번호는 입력할 수 없다.

다음과 같이 Caché 인스턴스를 위한 포트번호를 설정한다.:

- SuperServer 포트 번호: 1972 또는 위의 범위내에서 사용 가능한 숫자. 설치 후, 시스템 관리 포탈의 **[홈] > [환경 구성] > [메모리 및 시작]** 페이지에서 **[SuperServer 포트 번호]** 값을 변경할 수 있다.
- 웹 서버 포트 번호: 57772 또는 위의 범위내에서 사용 가능한 숫자. 설치 후, 시스템 관리 포탈의 **[홈] > [환경 구성] > [시작 설정]** 페이지에서 “WebServer 포트” 값을 변경할 수 있다.



[그림 2-12] 웹 서버 포트 변경

6. 초기 보안 설정 창에서는, 사용자가 초기 Caché의 보안 설정을 어느 정도로 할 것인지를 결정한다. [최소]를 선택하면, 다음 설치 작업으로 진행된다.

**참고:** 사용자가 최초 보안 설정으로 [최소]를 선택하는 경우, Caché에서 해당 드라이브와 프린터를 네트워크 상에서 사용이 필요하면, 서버 로컬 관리자 권한을 갖을 뿐만 아니라 Caché 서비스를 실행할 수 있는 원도우 계정을 새로 만들거나 기존 계정을 변경하여야 한다.

[보통]이나 [장금]을 선택하는 경우, Caché 서비스 실행을 위한 계정정보 입력창이 나타나는데 이 계정은 다음의 조건을 만족하여야 한다:

- 디폴트 시스템 계정: Windows 로컬 시스템 계정으로 Caché를 실행한다.
- 기존 원도우즈 사용자명 계정: %All 권한을 가진 Caché 계정을 생성하여, Caché 관리에 필요한 서비스들을 실행할 수 있도록 한다.

**참고:** 커베로스를 사용하는 경우, Caché 서비스를 실행할 수 있도록 정의된 사용자 계정을 입력해야 한다. InterSystems에서는 이러한 목적만을 위해 특별히 정의된 별도의 계정을 사용할 것을 권장한다.

‘다음’을 클릭하여, 입력된 계정에 대한 다음 사항들을 확인한다:

- 계정이 도메인상에 존재하는지 여부.
- 정확한 패스워드를 입력하였는지 여부.
- 제공된 계정이 서버상에서 로컬 관리자 권한을 갖는지 여부.

**주의:** Windows Vista에 설치한다면 계정을 입력하기 전에 로컬 관리자 권한을 가지고 있는지 확인하는 것이 필요하다. 현재로서는 설치 작업이 이 권한을 Vista 플랫폼에서 확인할 수 없기

때문이다.

7. 설치 준비 창에서, 지금까지 정의한 설치명(인스턴스명), 설치 유형, 설치 디렉터리를 확인한다.

**참고:** 라이센스 입력 창 – 설치 시스템에 Caché 라이센스 키가 설치되어 있지 않은 경우(즉 시스템 관리자 디렉터리, install\_dir\Wmgr, 에 Caché 키 파일이 없는 경우) 라이센스 입력창은 키가 없음을 표시하고 설치 중 라이센스 키를 입력할 것인지를 물어본다.

- 설치 중 라이센스 키를 입력하지 않고 계속 진행하려면, 선택 박스를 선택하지 않는다.  
라이센스 키는 설치가 완료된 후 시스템 관리 포탈의 **[홈] > [라이센스] > [라이센스 키]** 페이지를 통해 입력 가능하다.
- 설치 중 라이센스 키를 입력하는 경우, 선택 박스를 선택하여 Caché 라이센스 마법사를 실행하고 요구하는 라이센스 정보를 입력한다.

‘설치’를 클릭하면 지정된 설치 디렉터리에 Caché가 설치된다.

8. [설치]를 클릭하면 선택된 디렉터리에 Caché가 설치된다. 설치마법사 완료 대화 상자는 설치가 성공적으로 완료되었음을 사용자는 시작 페이지를 볼 것인지를 선택하고 마침을 클릭한다.

#### 4) 설치 후의 작업

설치된 Caché 인스턴스는 시스템 관리 포탈을 통해 관리한다. 시스템 관리 포탈에 대한 자세한 정보는 Caché 온라인 문서 Caché System Administration Guide의 “Using the System Management Portal”장을 참조하기 바란다.

- Caché의 다른 인스턴스를 원격으로 연결하려면, Caché System Administration Guide의 “Connecting to Remote Servers” 장의 “Define a Remote Server Connection” 부분에 설명된 절차를 따른다.
- Caché의 이전 버전에서 업그레이드하는 경우에는,
  - 오브젝트 재컴파일: 다음의 명령을 실행하여 전체 네임 스페이스내의 모든 Caché 오브젝트 어플리케이션을 다시 컴파일한다:

**Do \$system.OBJ.CompileAllNamespaces("u")**

**참조:** 이 메소드는 어플리케이션 뿐만 아니라 모든 클래스 정의를 업그레이드하고 재컴파일한다.

- 프록시(Proxy) 클래스 재생성: Caché 언어 바인딩을 통해 생성된 모든 프록시 클래스를 다시 생성해야 한다.
- 필요한 경우, 다음에서 설명하는 **추가 고려 사항**을 참고하여, 추가 보완 작업을 시행한다.

## 5) 추가 고려 사항

여기에서는 인증, 특정의 플랫폼, 설치 종류와 관련된 이슈, 태스크에 대해 설명한다:

- 하나 이상의 Caché 인스턴스 설치에 따른 이슈
- Caché 언어의 변경
- 설치된 Caché 인스턴스의 제거
- InterSystems Caché 패킷 드라이버

### 하나 이상의 Caché 인스턴스 설치에 대한 이슈

Caché 4.0 이상의 버전에서는 동일 Windows 시스템에 하나 이상의 인스턴스를 설치, 실행할 수 있는데 이런 경우 다음의 사항들을 고려하여야 한다:

첫째, Caché 인스턴스 각각에 고유한 이름, 설치 디렉터리, 포트 번호를 부여하여 독립된 시스템으로 구성한다. 그렇지만 복수의 인스턴스는 원도우즈 시스템상에서 하나 밖에 정의되지 않는 OS 기반 외부 리소스에 의해 제한을 받게 된다. 예를 들면, 일반적으로 원도우즈에는 하나의 웹 서버만 정의되어 있는데, 이 경우 마지막으로 설치된 Caché 버전의 CSP가 웹 서버에 최종적으로 구성하게 된다. 레지스트리에 저장된 Caché 클라이언트 구성 요소도 같은 문제를 갖고 있다. ODBC 드라이버와 ActiveX 컴포넌트들은 레지스트리에서 고유의 이름으로 저장되는데, 마지막 설치 버전의 컴포넌트들로 업데이트된다.

### Caché 언어의 변경

Caché 설치 시, 지원되는 모든 언어별 유ти리티 DLL들이 **install-dir\WBin** 디렉터리에 설치된다. 각 DLL은 해당 로컬 문자열과 메시지를 가지고 있으며, DLL 이름 형식은 **UTILaaa.DLL** 인데, 여기에서 aaa는 다음 언어를 나타내는 3 자리의 코드이다:

코드	언어
CSH	중국어(간체)
DEU	독일어(표준)
ENU	영어(미국)
ESP	스페인어(스페인)
FRA	프랑스어
ITA	이태리어(표준)
JPN	일본어
KOR	한국어
NLD	네덜란드어(표준)
PTB	포르투칼어 (브라질)
RUS	러시아어

### [표 2-3] Caché 언어

Caché 설치 로케일은 다음과 같이 ^NLS 루틴을 사용하여 변경할 수 있다:

```
%SYS>Do ^NLS

1) Locale definitions
2) Table definitions
3) Current system settings

NLS option? 1

Select a locale: enuw => English, United States, Unicode

1) Display locale
2) Edit locale
3) Install locale
4) Export locale
5) Import locale
6) Validate locale
7) Copy locale
8) Delete locale
9) Load locale table
10) Select another locale

Locale option? 10

Select a locale: enuw => itaw - Italian, Italy, Unicode
```

8 비트 로케일간 또는 유니코드 로케일 사이에서만 변경 가능하며, 8 비트에서 유니코드나 또는 그 반대로의 변경은 허용되지 않는다.

### Caché 제거

Caché를 제거하려면, 우선 Caché를 중지하고, 그 다음에 Caché 큐브를 윈도우즈 작업 표시줄의 알림 영역에서 삭제한다 – 작업 표시줄에서 Caché 큐브를 없애려면 Caché 큐브 아이콘 메뉴에서 “나가기(x)”를 선택한다.

Caché를 제거하기 위해서는, Windows 제어판의 프로그램 추가/제거 어플리케이션을 통해 액세스할 수 있는 Caché 제공 제거 프로그램만을 사용하여야 한다.

**참고:** 지원되지 않는 다른 제거 프로그램 사용하는 경우, 불완전한 제거를 야기할 수 있으므로 예상치 못한 결과를 초래할 수도 있다.

## InterSystems Caché 패킷 드라이버

### Windows 2000, XP, 2003 서버에 Caché 패킷 드라이버 설치하기

윈도우즈 시스템(Windows 2000, XP, 및 2003 서버 시스템)에서 Caché 와 함께 Raw Ethernet, DDP, 또는 LAT를 사용하려면, 다음과 같이 적합한 패킷 드라이버를 설치해야 한다:

1. 데스크톱의 내 네트워크 환경에서 속성을 클릭한다. 시작을 클릭하고, 설정에서 네트워크 및 Dial-up 연결을 선택하여도 된다.
2. 로컬 영역 연결의 속성을 선택한다.
3. 설치를 선택한다.
4. 네트워크 구성요소 유형 선택 대화 상자에서 프로토콜을 클릭한 후 추가를 클릭한다.
5. 네트워크 프로토콜 선택 대화 상자에서 디스크 있음을 클릭한다.
6. 패킷 드라이버 파일 경로를 입력한 후 확인을 클릭한다. 또는 검색 버튼을 사용하여 `ispkt2k.inf` 파일을 가지고 있는 경로인 `\drivers\win2k`를 검색할 수도 있다. 열기를 선택하고, 확인을 클릭한다.
7. 운영 시스템에 적합한 드라이버를 선택한 후 – 예를 들어, InterSystems Packet Driver for Windows 2000, XP and 2003 서버 – 확인을 누른다.
8. 드라이버가 설치되면 닫기를 클릭한다.

Windows를 다시 시작하면, 변경된 설정이 Caché에 적용된다.

## 2. Caché Studio 소개

Caché Studio는 응용프로그램을 빠르고 효율적으로 개발할 수 있는 통합된 단일 개발 환경으로서 다음과 같은 기능들을 제공한다:

- 데이터베이스 클래스, 웹 서비스 클래스등과 같은 클래스 정의
- CSP (Caché Server Pages), Zen, XML, HTML, Javascript, Cascading Style Sheets (CSS) 를 사용하는 대화형 웹 페이지 정의
- Caché ObjectScript, Basic, MultiValue 등을 사용한 프로그램 루틴
- Caché ObjectScript, Basic, Java, SQL, JavaScript, HTML, XML에 대한 통합 구문 체크
- 응용프로그램 소스의 공동 관리를 위한 소스 공유 저장소 지원
- 윈도우 기반 소스 코드 디버깅
- 프로젝트를 통한 응용프로그램 소스 코드 관리 기능

Caché Studio는 Windows 운영 체계에서 실행되는 Caché 오브젝트 기반의 클라이언트 응용프로그램이다. 서버 운영 환경과 상관없이, 현재의 Caché Studio 버전과 호환되는 모든 Caché 서버와 연결된다.

**참고:** Studio 클라이언트 버전은 연결하고자 하는 Caché 서버와 동일하거나 상위 버전이어야 한다.

예를 들면, Caché 2007.1 Studio는 2007.1 Studio 서버나 그 이전 버전에 연결되지만 Caché 2007.2 서버나 그 이후 버전에는 연결이 되지 않는다.

## 1) Studio Window 개요

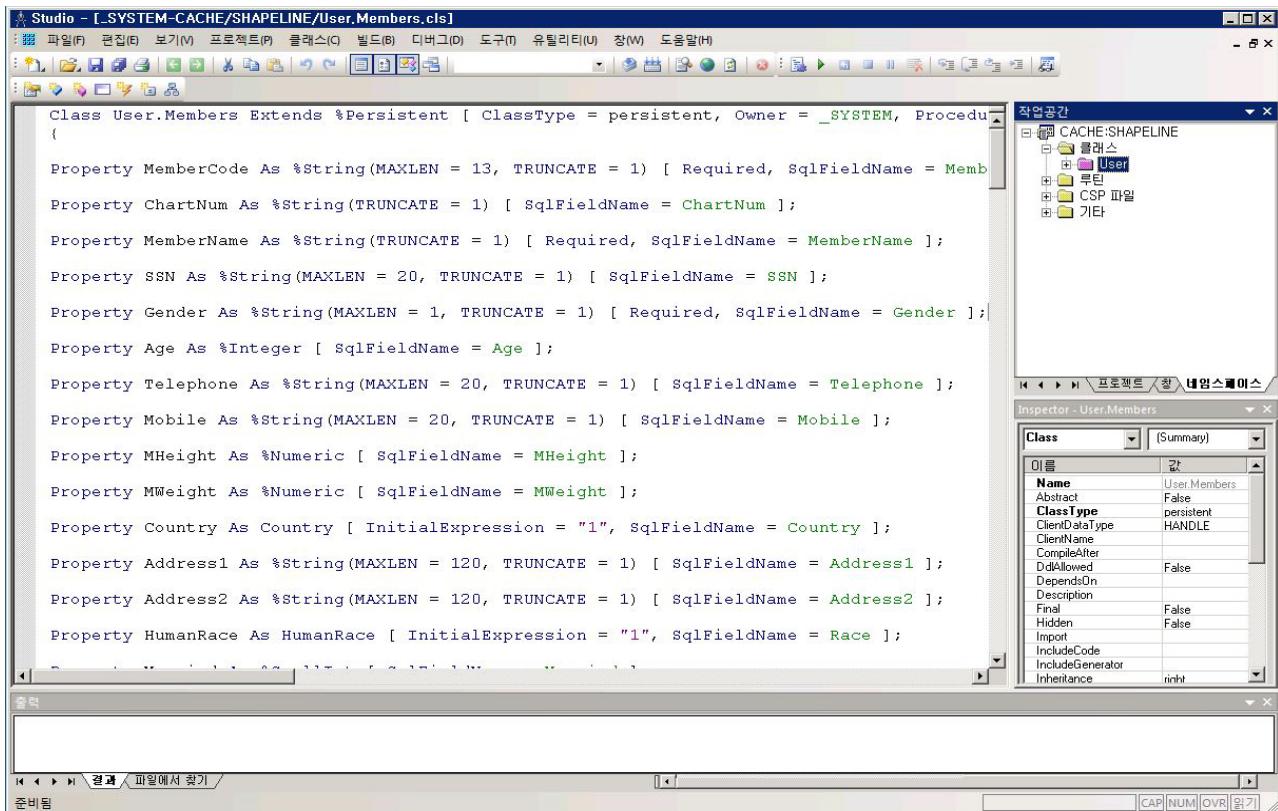
Caché Studio는 Windows 응용프로그램으로 편집창을 사용하여 편집 대상의 속성을 표시하고 편집한다.

Caché Studio 사용자 인터페이스의 주요 구성 요소는 다음과 같다:

- **편집기:** 클래스 정의를 편집하는 클래스 편집기, 루틴을 편집하는 루틴 편집기, CSP 정의 텍스트를 편집하는 CSP 편집기로 이루어진다.
- **작업 공간 창:** 세 개의 탭으로 이루어져 있는데 현재 **프로젝트**의 컨텐츠, 현재 오픈되어 있는 모든 파일들을 갖고 있는 **창**, 현재 Studio가 연결하고 있는 **네임스페이스**의 컨텐츠 등이다.
- **인스펙터 창:** 현재 오픈되어 있는 클래스 정의 속성들을 키워드를 확인하고 수정한다.
- **출력 창:** 컴파일 메시지들과 같이 Caché 서버로부터 생성되는 출력 메시지를 표시한다.

위에 설명된 창들 외에도, Caché Studio 공통적인 작업을 지원하는 다음과 같은 마법사와 템플릿등을 포함하고 있다:

- **클래스 마법사:** 새로운 클래스를 정의한다.
- **클래스 멤버 마법사:** 프로퍼티, 인덱스, 관계, 메소드, 파라미터, SQL 트리거, 쿼리, 프로젝션, 저장, 외래 키, XData 블록등 클래스 정의에 포함되는 구성 요소들을 정의, 추가한다.
- **외부 오브젝트들에 대한 프락시 클래스를 생성하는 마법사:** 즉, Java 클래스와 jar 파일, SML 스키마, SOAP 클라이언트 클래스에 대응하는 Caché 클래스를 생성할 수 있으며 COM 객체 및 .NET의 DLL 어셈블리 파일에 접근할 수 있는 기능을 제공한다.
- **HTML 템플릿:** 색, 표, 태그 스크립트를 더 해준다.
- **CSP 품 마법사:** CSP 페이지에서 Caché 오브젝트와 결합된 HTML 양식을 자동 생성한다.
- **Zen 템플릿:** 도표, 표, 메소드, 스타일을 추가한다.



[그림 2-13] Caché Studio

## 2) 프로젝트

Caché Studio에서는 프로젝트를 사용하여, 응용프로그램 소스 코드를 관리한다. 프로젝트는 클래스 정의, CSP 파일, 루틴의 집합체를 말하는데, 단일 어플리케이션에 필요한 모든 클래스 및 CSP 파일들을 하나의 프로젝트에 모두 등록시켜 체계적으로 분류, 관리할 수 있다. 즉, 하나의 프로젝트를 오픈하면 주어진 어플리케이션에 관련된 모든 컴포넌트들이 자동적으로 로드된다.

Studio에 로그인하게 되면, 사용자는 사용자 정의 프로젝트나 Studio 디폴트 프로젝트 (Default\_yourusername(Default\_접두어 다음에는 사용자 이름)) 중 하나의 프로젝트에서 작업을 하게 된다.

단일 프로젝트에서 모든 파일은 같은 네임스페이스와 Caché 서버에 있어야 한다. 각각의 클래스, CSP 파일, 루틴은 동시에 여러 프로젝트에 포함될 수 있으며, 하나의 네임스페이스는 여러개의 프로젝트를 가질 수 있다.

프로젝트에는 클래스나 CSP 파일을 편집할 때 사용하는 주어진 네임스페이스에서의 클래스 계층 정보등이 저장되며, 디버깅 응용프로그램을 시작하는 방법등 디버깅 정보도 갖고 있다.

### 3) 클래스 정의

메뉴 ‘파일 → 새로만들기 → 일반 탭’에서의 “Cache 클래스 정의”를 선택하여 Caché 클래스를 정의한다. 클래스 정의는 클래스 구성 요소(프로퍼티, 메소드 등)와 키워드(keywords) 등과 같은 항목들로 구성되어 있는데, 그것들은 각각의 설정값과 연결되어 클래스의 기능을 구체적으로 명시한다.

클래스 정의는 클래스 사전(Class Dictionary)에 저장되며, 이는 Caché 데이터베이스에 보관된다. 클래스 정의는 컴파일 과정을 통하여 정의된 클래스의 오브젝트 인스턴스를 만드는 실행 코드를 생성한다. 클래스를 위해 만들어지는 실행 코드의 소스 코드는 한 개 이상의 Caché 루틴으로 구성된다. 사용자는 이렇게 만들어진 루틴을 Caché Studio에서 볼 수 있다.

정의된 클래스는 다른 테크놀로지 환경에서도 사용될 수 있는데 SQL과 ActiveX의 경우에는 자동적으로 즉시 사용 가능하며, Java 및 C++에서는 추가적인 컴파일 절차를 거쳐 Caché 클래스에 대응하는 Java 클래스를 생성한다.

Caché Studio에서의 클래스 정의는 클래스 편집창에서 편집된다. 또한 클래스는 인스펙터 창을 통해 정의된 자신의 키워드 및 속성들을 볼 수 있다.

#### 클래스 정의 코드

다음은 한 개의 프로퍼티와 한 개의 메소드를 가지고 있는 클래스를 정의하는 예를 보여준다:

```
/// Definition of a simple persistent Person class
Class MyApp.Person Extends %Persistent [ClassType = persistent]
{
    /// The name of the Person
    Property Name As %String;

    /// A simple Print method
    Method Print() As %Boolean
    {
        // Write out Person's Name
        Write "Name is: ", ..Name
        Quit 1
    }
}
```

#### 클래스 정보

클래스 정의는 클래스 이름과 속성을 다음과 같이 선언함으로서 시작한다:

```
Class MyApp.Student Extends Person
{
```

```
}
```

여기에서 MyApp.Student 라고 불리는 클래스가 정의된다(아직은 프로퍼티, 메소드, 또는 다른 구성 요소들이 정의되지 이전이다). 이 클래스는 MyApp.Person 클래스를 확장한 것으로서 (Student 와 Person이 같은 패키지로 되어 있으면 확장 명령문에서는 MyApp가 생략될 수 있다) {} 내에 클래스 멤버를 정의한다.

클래스 자체에 대한 추가적인 속성은 클래스 선언 다음에 대괄호([])를 이용하여 설정할 수 있다. 예를 들면, 생성하는 클래스를 Final로, 클래스에 대응되는 SQL 테이블명은 StudentTable로 정의할 경우 다음과 같이 작성된다:

```
Class MyApp.Student Extends Person [Final, SqlTableName=StudentTable]
{
}
```

또한 클래스 선언 이전에 바로 설명 주석을 달아 (3 개의 사선으로 표시 ///) 이 클래스에 대해 설명을 제공할 수도 있다. 이러한 설명은 Caché Online Class Reference 를 통해 해당 클래스 문서를 참조할 때 볼 수 있다. 또한 설명에는 HTML 마크업을 사용할 수도 있다:

```
/// This is a simple Student class
/// It is derived from the Person class
Class MyApp.Student Extends Person
{
```

// (2 개의 사선) 또는 /\* \*/ (사선, 별표로 시작하고, 별표, 사선으로 끝냄) 를 사용하여 C 스타일의 코멘트 모양을 구현할 수도 있으며, 클래스 정의 어느 곳에서나 일부 내용을 코멘트로 뺄 수 있다.

## 프로퍼티

클래스 정의에서는 다음과 같이 **Property** 키워드를 사용하여, 클래스 프로퍼티를 정의한다:

```
Class MyApp.Student Extends Person
{
    Property GPA As %Float;
}
```

위의 예는 %Float 타입(As를 사용하여 설정)의 프로퍼티명 GPA 를 갖는 프로퍼티를 정의하고 있다. 프로퍼티 정의의 끝은 세미콜론(:)으로 표시된다. 클래스 선언과 마찬가지로 이 프로퍼티에 대한 설명을 선언 앞에 /// 주석을 사용하여 추가할 수 있으며, 추가 프로퍼티 키워드를 [ ] 괄호에 명시한다.

```
Class MyApp.Student Extends Person
{
    /// Grade Point Average for the Student
    Property GPA As %Float [ Required ];
}
```

프로퍼티 타입의 파라미터 값을 미리 정의하려면 (파라미터는 프로퍼티 기능을 추가로 정의할 수 있게 해줌) 그것을 타입의 일부로서 ( ) 괄호안에 넣는다. 파라미터 값은 문자값으로 취급되므로, 문자열의 경우 인용부호 안에 넣는다.

```
Class MyApp.Student Extends Person
{
    /// Grade Point Average for the Student
    Property GPA As %Float(MINVAL=0.0, MAXVAL=5.0) [ Required ];
}
```

## 메소드

클래스 정의에서는 다음과 같이 **Method** 키워드를 사용하여, 메소드를 정의한다:

```
Class MyApp.Student Extends Person
{
    /// This method wastes <varname>count</varname> seconds.
    Method WasteTime(count As %Integer=1) As %Boolean [Final]
    {
        // loop and sleep
        For i = 1:1:count {
            Write "."
            Hang 1
        }
        Quit 1
    }
}
```

메소드의 리턴 타입은 메소드 이름 다음에 **As** 를 사용하여 명시한다. 파라미터명은 메소드명 다음에 오는 ( ) 괄호에 넣으며, 메소드 코드는 메소드 선언 다음에 나오는 { } 괄호에 넣는다. 프로퍼티와 마찬가지로 /// (3 개의 사선)을 사용하여 메소드에 대한 설명을 제공한다. 필요하면 HTML 마크업을 사용할 수 있으며, 추가 키워드 값은 [ ] 괄호 안에 넣는다.

메소드의 프로그래밍 언어는 **Language** 키워드를 사용하여 정의한다. 예를 들면, 아래의 코드는 **Basic** 으로 메소드를 정의하고 있다:

```
/// Find the sum of numbers from 1 to <var>count</var>.
Method SumUp(count As %Integer) As %Integer [Language = basic]
{
    total = 0
    For i = 1 To count
        total = total + i
    Next

    Return i
}
```

**Language** 키워드값은 다음 언어 중 하나를 설정한다:

- **cache** : Caché ObjectScript (디폴트 값). 자세한 내용은 Caché 온라인 문서 Using Caché ObjectScript 를 참조하기 바란다.
- **basic** : Basic. Caché 에서는 Basic 프로그래밍 언어를 지원한다. (Caché ObjectScript와 같은 방법으로) Basic 메소드는 Caché 가상 머신에서 실행 가능한 코드로 컴파일 된다. 자세한 내용은 Caché 온라인 문서 Using Basic 을 참조하기 바란다.
- **java** : Java. Caché Java Binding 사용시, 자바 메소드는 자동 생성된 자바 클래스의 일부분이 되어 실행 가능한 자바 코드로 컴파일된다. 자세한 내용은 Caché 온라인 문서 Using Java with Caché 를 참조하기 바란다.

단일 클래스에 서로 다른 언어를 사용하는 복수의 메소드를 가질 수도 있고 클래스 레벨에서 **Language** 키워드를 사용하여 전체 클래스에 대한 기본 프로그래밍 언어를 명시할 수도 있다.

#### 4) CSP 파일

CSP (Caché Server Page) 파일은 CSP 마크업 언어를 포함하는 HTML 또는 XML 텍스트 파일을 말한다. CSP 엔진은 CSP 파일을 컴파일하여 Caché 클래스를 생성하여 HTTP 이벤트에 응답하도록 웹 컨텐츠를 구성한다.

웹 개발에 있어서 좀더 프로그램적 요소를 추가하려면 클래스를 정의하는 동일한 방식으로 Caché Studio를 이용하여 CSP 클래스를 생성하고 수정할 수 있다. Caché Studio에서는 CSP 파일을 CSP 편집기 창에 표시하며, 이 편집기는 HTML과 XML의 구문 검증과 CSP 파일에 포함되는 많은 스크립트 언어를 제공한다.

CSP 편집기에서는 CSP 마크업 태그를 삽입하는 것과 같은 일반적인 CSP와 HTML 편집 업무를 실행하는 명령어가 제공된다. 그리고 Studio 메뉴 보기 -> 웹 페이지를 선택하여, 브라우저에서 CSP 파일의 결과를 바로 확인할 수가 있다.

#### 5) 루틴 편집기

메뉴 ‘파일 -> 새로 만들기 -> 일반 탭’에서 (Cache ObjectScript/Cache Basic) 루틴 아이콘을 선택하여 Caché 루틴을 생성하고, 편집한다.

#### 6) 복수의 사용자 지원

Caché Studio는 오브젝트 기반의 클라이언트/서버 응용프로그램이다. Studio에서 생성하고 편집하는 소스 파일(클래스 정의, 루틴, CSP 파일)은 Caché 서버에 저장되며, 오브젝트로 표현된다.

Studio에서 소스 파일은 현재 연결되어 있는 Caché 서버에 저장된다. 만약 소스 파일이 오픈되어 있는 상태에서 수정하기 전 다른 사용자에 의해 동시에 오픈되고 먼저 수정이 이루어졌으면, 수정된 새로운

버전을 로드할 것인지 여부를 Studio는 물어보게 된다.

Studio는 여러 사용자들이 동일한 소스를 동시에 보고 있는지를 자동으로 탐지하여 동시 접속을 관리한다. 다른 사용자가 편집하고 있는 파일을 열려고 하는 경우, 컴퓨터는 읽기 전용으로 열 것인지를 물어보는 것 등이 이런 예에 해당한다.

## 7) Caché 문서 가져오기 / 내보내기

Studio에서 편집 중 생성되는 (클래스 정의, 루틴 등) 모든 소스는 Caché 데이터베이스에 저장되며, Studio 메뉴 ‘도구’의 가져오기/내보내기 메뉴들을 사용하여, 파일로 가져오거나 내보낼 수 있으며 선택된 클래스 및 루틴은 XML 문서로 파일에 저장된다.

## 8) 디버깅

Caché Studio는 소스 레벨의 GUI 디버거를 제공한다. 이 디버거를 통하여 Studio와 연결되는 Caché 서버에서 대상 프로세스를 추가하거나 실행시킨다. 디버거는 대상 프로세스를 원격 조정하며, 사용자는 변수를 확인하고, 단계별로 코드를 실행하고, 중지점을 설정할 수 있다.

사용자가 디버거를 사용하려면 프로젝트를 사용 하여야 하는데, 프로젝트에는 디버깅에 필요한 정보(루틴 이름, 메소드, CSP 페이지, Zen 페이지, 클라이언트 응용프로그램)가 포함되어 있기 때문이다. 또한 프로젝트에는 재사용이 가능하도록 이전의 디버깅 세션에서 설정된 중지점 목록이 저장되어 있다.

프로젝트를 사용하지 않고 실행중인 프로세스에 중지점을 사용 디버깅할 수도 있지만, 이러한 경우 Studio는 이전 세션에 설정된 중지점을 기억하지 못한다.

### 오브젝트 기반의 응용프로그램 디버깅

여기에서 Caché Studio는 INT(Caché ObjectScript 루틴)과 BAS (Basic 루틴) 파일의 소스 레벨 디버깅만을 할 수 있다. 코드에 들어가거나 클래스나 CSP 페이지에 중지점을 설정하려면, 상응하는 INT나 BAS 파일을 열어, 거기에 있는 디버깅 명령어를 사용한다. 해당 클래스로부터 생성되는 소스 코드를 사용하려면, Studio 도구 -> 옵션 메뉴의 옵션창에서 컴파일러->플래그&최적화 항목의 ‘플래그 컴파일’ 부분에서 ‘생성된 소스 코드 유지’를 선택한 후 컴파일한다.

## 9) Caché 보안 시스템과의 통합

Caché 보안 시스템은 Studio의 사용 및 서버와의 연결을 제어한다. Studio를 시작하면 로그인 화면이 나오는데, 최소한 다음과 같은 권한을 갖는 사용자로 로그인 하여야 한다:

- **%Development:Use** – 다양한 개발 관련 리소스를 사용할 수 있는 %Development 리소스에 대한 사용 허가.
- **%Service\_Object:Use** – Studio 연결을 제어하는 %Service\_Bindings 서비스를 사용할 수

있는 **%Service\_Object** 리소스에 대한 사용 허가.

네임스페이스의 기본 데이터베이스에 대한 읽기 또는 쓰기 허가를 가진 경우에만, 사용자는 해당 네임스페이스에 연결된다. 이러한 종류의 권한이 사용자에게 어떻게 허용되는지는, 인스턴스의 보안 수준에 의해 결정된다.

- 최소 보안의 인스턴스: UnknownUser를 포함한 모든 사용자는 모든 네임스페이스에 대해 모든 사용 권한을 가진다. Studio 로그인 화면이 나오면 사용자명, 패스워드를 빈 칸으로 남기거나, 또는 각 칸에 각각 "\_SYSTEM", "SYS" 를 입력한다.
- 보통 보안의 인스턴스: 사용자는 Studio 사용에 필요한 특정 권한을 명시적으로 허가 받아야 한다. 이러한 권한들을 포함하는 역할을 부여 받아 허용된다.
- 잠금 보안의 인스턴스: Studio 접근을 제어하는 서비스(%Service\_Bindings)가 기본값으로 '사용가능' 하지 않다. 즉, Studio를 사용할 수 없다.

참고: Studio 사용은 설치 후에 발생하는 기본 설정의 변경 사항에 따라서도 영향을 받을 수 있다.

## 10) 소스 제어 흑(Hook)

Studio에는 사용자 흑(Hook)을 구현하는 메커니즘이 있는데, 흑(hook)은 문서가 로딩되거나 저장될 때, Caché 서버에서 실행되는 코드를 말한다. 일반적으로 이러한 흑은 소스 또는 변경 제어 시스템으로의 연결을 구현하는 데 사용된다.

소스 제어 흑을 정의하려면, **%Studio.SourceControl.Base** 클래스의 하위 클래스를 생성하여 원하는 콜백 메소드를 실행한다. Studio에서 어떤 소스 제어를 사용할 것인지 명시하려면, 시스템 관리 포탈의 [흡] > [환경 구성] > [소스 컨트롤 설정] 페이지에서 정의한다. 자세한 설명은 **%Studio.SourceControl.Base** 클래스나 Caché 온라인 문서 “Using Caché Studio” 의 Using Studio Source Control Hooks 부록을 참조하기 바란다.

## 3. Caché 터미널

Caché 터미널은 Caché 명령어를 실행하고, 현재 정의되어 있거나 저장된 데이터를 볼 수 있는 간단한 명령어 라인 인터페이스를 말한다. 사용자의 초기 학습, 개발 및 디버깅 시에 유용하게 사용할 수 있다.

### 1) Caché 터미널 시작하기

Caché 터미널은 대화형 또는 배치 모드로 사용할 수 있다.

터미널을 대화형으로 사용하려면, 다음 중 한 가지 방식을 사용한다:

- 로컬 데이터베이스를 사용하는 터미널: Caché 큐브에서 터미널을 선택한다.

- 원격 서버에 있는 데이터베이스를 사용하는 터미널: Caché 큐브에서 [원격 시스템 액세스] > [터미널]을 통해 서버 이름을 선택한다. 또는 Caché 큐브의 [원격 시스템 액세스] > [Caché 텔넷]을 선택한 후 ‘연결’ 메뉴에서 ‘호스트’를 선택, 원격 시스템의 ip 및 포트 번호를 입력하거나 ‘연결’ 메뉴에 리스트되어 있는 서버를 선택한다. 필요한 경우, 원격 시스템에 액세스하기 위한 OS 로그인 계정이 필요하며, OS 로그인 후 Caché 사용자 이름과 패스워드를 사용하여 Caché 시스템에 로그인한다. 추가 옵션 및 보다 자세한 정보는 Caché 온라인 문서 Caché System Administration Guide의 Connecting to Remote Servers를 참조하기 바란다.

이러한 절차를 통하여 터미널 창에 로그인이 되는데, 창에 표시되는 프롬프트는 사용자가 현재 작업중인 네임스페이스를 나타낸다. 예를 들어,

```
USER>
```

는 현재 “USER” 네임스페이스에 있음을 나타낸다.

## 2) 특성

Caché 터미널은 Caché 응용프로그램과 호환되도록 설계되었으며, Digital Equipment Corporation VT320 Terminal의 기능을 많은 부분 모방하고 있다. Caché 터미널에서는 두 가지 메소드를 사용, Caché 로컬 시스템 및 네트워크와 통신하는데, 제록 표시줄은 현재 사용 중인 통신 모드를 나타낸다.

- 로컬 통신은 Caché 터미널이 로컬에 설치된 Caché 서버와 통신할 때 사용되며, 이 경우 제록 표시줄은 Cache TRM:pid(configname)로 나타난다. 여기에서
  - pid는 Caché 터미널이 통신하는 Caché 프로세스의 ID이며,
  - configname은 Caché 서버 프로세스가 시행되고 있는 Caché의 인스턴스명을 말한다.
- 네트워크 통신은 TCP/IP 기반의 TELNET 프로토콜을 사용하여, Windows Caché 서버나 UNIX 또는 VMS 호스트와 통신한다. 이 경우 제록 표시줄은 (server — Caché Telnet)로 나타나는데, 여기에서 server는 원격 서버의 호스트명을 의미한다.

Caché의 통신 스택은 Winsock 인데, 그것은 BSD(Berkeley Software Distribution)에서 대중화 시킨 소켓 파라다임에 기반한 Microsoft Windows의 네트워크 프로그램 인터페이스이다. 호스트 입력은 로컬 파일, IP 주소, 또는 Winsock 구현 방식이 네임 서버를 사용할 수 있는 경우 일반적인 호스트명이 된다. 호스트 이름 다음에는 선택적으로 #nnn을 사용하여, 비표준 포트 번호를 명시할 수 있다.

이러한 통신 모드에서 보고되는 오류는 Winsock 오류 코드명으로 예를 들어, WSAECONNREFUSED는 연결이 거절되었음을 의미한다.

### 3) 사용 용도

Caché 터미널에서는 모든 종류의 Caché 명령어를 실행할 수 있다. 예를 들면,

```
d ^myroutine
```

```
set dirname = "c:\test"
```

```
set obj=##class(Test.MyClass).%New()  
write obj.Prop1
```

**참고:** Caché 터미널에서는 사용자가 입력하는 각 줄마다 암시적으로 Use 0 라는 명령어가 실행하는데, 다른 디바이스에 출력을 지시하는 Use 명령어를 사용되면, Use 0 는 무시된다. 또한 대량 입력 버퍼의 경우 Ctrl-c, Ctrl-s 처럼 입력 흐름을 정지시키는 키의 실행이 지연될 수 있다. 이러한 지연은 프로세서와 연결 속도에 따라서도 좌우된다.

사용자 파일 시스템에 있는 .scr 를 확장자로 갖는 터미널 스크립트 파일을 실행할 수도 있다. Caché 터미널에는 이 스크립트에 사용할 수 있는 일단의 명령어가 제공되는데, 사용자가 직접 입력하는 것처럼 Caché 명령어를 터미널로 보내는 명령어등을 포함한다.

### 4) ZWELCOME 루틴

Caché 터미널이 실행되기 시작하면, 코드는 %SYS 네임 스페이스에서 ZWELCOME라는 루틴이 있는지 확인한다. 그러한 루틴이 있으면, 터미널 로그인 전에 이것을 먼저 호출한다. 루틴명이 시사하듯이 사용자에게 고유의 식별 방식과 환영 메시지를 터미널에 출력하는데 사용된다.

**참고:** ZWELCOME을 %SYS 네임 스페이스로 설치하려면 관리자 권한과 CACHESYS 데이터베이스 쓰기 접근을 가진 사용자여야 한다.

**주의:** ZWELCOME 루틴은 \$USERNAME은 설정되지 않고 %ALL 권한을 갖는 \$ROLES에 의해 %SYS 네임스페이스에서 실행된다. ZWELCOME에서 오류가 발생하지 않도록 주의하여야 한다.

**예제:**

```
ZWELCOME() PUBLIC ;  
; Example  
Write !  
  
Set ME = ##class(%SYS.ProcessQuery).%OpenId($JOB)  
  
Write "Now: ", $ZDATETIME($HOROLOG, 3, 1), !  
Write "Pid/JobNo: ", ME.Pid, "/", ME.JobNumber, !  
Write "Priority: ", ME.Priority, !  
  
Quit
```

## 5) 시작 네임스페이스

Caché 터미널을 처음 시작하면, 특정 네임 스페이스에 속하게 되는데 이 시작 네임스페이스는 Caché 시스템 관리 포탈의 [홈] > [보안 관리] > [사용자] > [사용자 편집] 페이지에서 ‘시작 네임스페이스’란에서 선택할 수 있다. 설정이 되어 있지 않은 경우, 디폴트 값은 ‘USER’ 네임스페이스이다.

Caché 터미널 프롬프트에는 “**USER>**” 과 같이 현재의 네임스페이스가 표시된다.

### 네임스페이스 변경

Caché 터미널에서 다음과 같은 방법으로 네임스페이스를 바꿀 수 있다:

- ZNSPACE 명령
- ^%CD 유ти리티
- \$ZU(5) 함수

#### ➤ ZNSPACE 명령

ZN(ZNSPACE의 약어) 명령어를 통해 새로운 네임스페이스로 변경한다:

```
USER>ZN "SAMPLES"  
SAMPLES>
```

틀린 네임스페이스 이름을 입력하면, ZNSPACE는 <NAMESPACE> 오류를 리턴한다.

#### ➤ %CD 유ти리티

^%CD (change directory) 유ти리티를 사용하여, 네임스페이스를 바꿀 수도 있다:

```
SAMPLES>d ^%CD  
  
Namespace: user  
You're in namespace USER  
Default directory is c:\intersystems\cache\mgr\user\  
USER>
```

여기에서 ^%CD 유ти리티는 바꾸려 하는 새로운 네임스페이스를 입력값으로 받아 변경한다. 위의 예처럼, 네임스페이스명은 대소문자를 구별하지 않는다. 프롬프트에서 “?”를 입력하면, ^%CD에는 사용 가능한 네임스페이스 목록이 표시된다. 틀린 네임스페이스를 입력하면, ^%CD에는 오류 메시지가 리턴된다.

#### ➤ \$ZUTIL(5) 함수

\$ZU(5)(\$ZUTIL(5)의 약어) 함수를 사용하여 새로운 네임스페이스로 변경한다:

```
USER>Do $ZUtil(5,"Samples")  
SAMPLES>
```

이 함수는 두 번째 인자에 바꾸려 하는 네임스페이스의 이름을 가진다. 위와 같이, 네임스페이스는 대소문자를 구별하지 않는다. 틀린 네임스페이스를 입력하면 \$ZUTIL(5)는 <NAMESPACE> 오류를 리턴한다.

## 6) 터미널 프롬프트

앞에서 설명한 것처럼, 터미널 프롬프트에는 사용자가 현재 진행중인 네임스페이스를 표시한다. 트랜잭션 레벨 또는 프로그램 스택 레벨을 나타내는 추가 정보를 표시할 수도 있다.

### 트랜잭션 레벨

트랜잭션 중이면, 접두어로 트랜잭션 레벨을 나타낸다. 접두어는 TLn 형식인데, 여기에서 n은 트랜잭션 레벨을 말한다. 예를 들어, 사용자가 User 네임스페이스에 있으면서 Caché ObjectScript 명령어 TSTART를 입력하면, 프롬프트는 다음과 같이 변경된다.

```
USER>tstart
```

```
TL1:USER>
```

터미널에서 나오면, 트랜잭션은 룰백한다.

### 프로그램 레벨

오류가 발생하면, 다음과 같이 프롬프트뒤에 프로그램 스택 레벨이 표시되어 디버깅 모드로 변환된다.

```
USER 5d3>
```

Quit 명령어를 입력하여 디버깅 모드를 종료하거나 오류를 디버깅할 수 있다. 기타 자세한 내용은 Caché 온라인 문서 “Using Caché ObjectScript”의 Command-line Routine Debugging 장을 참조하기 바란다.

### TSQL 쉘(shell)

TSQL 쉘에 접근하려면 DO \$SYSTEM.SQL.TSQLShell()를 입력한다. 그러면 프롬프트는 :TSQL 문자열을 가지고 다음과 같이 표시된다.

```
USER>DO $SYSTEM.SQL.TSQLShell()
```

```
Current settings :-
```

```
No current settings
```

```
Compiler is NEW
```

```
USER:TSQL>
```

TSQL 쉘에서 나오려면 ^ 명령어를 입력한다.

자세한 내용은 Caché 온라인 문서 “Caché Transact-SQL (TSQL) Migration Guide” 의 Using the TSQL Shell 장을 참조하기 바란다.

## 운영 시스템 쉘

Caché 터미널에서는 또한 다양한 운영 시스템 쉘을 사용할 수 있다. 그렇게 하기 위해서는 ! 를 입력한다. 그러면 기본 운영 시스템 쉘이 나오고 프롬프트에는 수행할 디렉터리가 다음과 같이 표시된다.

```
USER>!  
c:\Wintersystems\cache\mgr\user\>
```

**참고:** Macintosh에서는 C-shell을 이러한 방법으로 열 수 없어, 허가 거부 오류가 나온다. 그러나 다른 쉘(Bash, Bourne, Korn)은 사용할 수 있다. 쉘에서 나오려면, 쉘에 따라 quit 나 exit 명령어를 사용한다.

## 7) 터미널에서 나오기

Caché 터미널을 종료하는 방법에는 다음과 같이 3 가지 방법이 있다:

- ‘HALT’ 명령을 실행한다.
- 터미널 메뉴 ‘파일 > 종료’를 선택한다.
- Alt+F4 를 누른다

이 때, 사용중인 파일이 있으면 모두 닫은 후에, 터미널은 종료된다. 터미널이 시작시 서버에 연결되어 있으면 통신 채널이 종료되며 터미널도 같이 종료된다.

Cache 텔넷을 통하여 터미널에 접근한 경우에는, 통신 채널이 종료되면 터미널이 자동으로 종료되지 않기 때문에 ‘연결’ 메뉴를 통해 다시 연결할 수 있다.

# 제 3 장 Caché ObjectScript

## 1. Caché ObjectScript 소개

- 1) 특징
- 2) 언어 개요
- 3) 명령어, 함수의 호출
- 4) ANSI 표준 M과의 관계

## 2. 변수

- 1) 변수의 범주
- 2) 변수 타입 정하기와 전환
- 3) 변수 선언과 범위

## 3. 연산자

## 4. 명령어

- 1) 명령어 인수
- 2) 명령어의 후조건(postconditional) 표현식
- 3) 코드 호출
- 4) 지정 명령어
- 5) 흐름 제어 명령어
- 6) 입출력 명령어
- 7) 기타 명령어

## 5. 트랜잭션 처리

- 1) 응용프로그램에서의 트랜잭션 관리
- 2) 자동 트랜잭션 롤백
- 3) 시스템 레벨에서의 트랜잭션 이슈

## 1. Caché ObjectScript 소개

Caché ObjectScript 은 빠르게 변화하는 다양한 비즈니스 어플리케이션 개발에 적합하게 설계된 오브젝트 프로그래밍 언어로서, 비즈니스 로직, 어플리케이션 통합, 데이터 처리등의 다양한 응용프로그램에 적합하다.

Caché ObjectScript 소스 코드는 Caché Virtual Machine 에서 실행되는 오브젝트 코드로 컴파일된다. 이 컴파일된 오브젝트 코드는 스트링처리와 데이터베이스 액세스등을 포함한 전형적인 비즈니스 어플리케이션 작업들을 위하여 최적화되어 있다. Caché ObjectScript 프로그램은 Caché가 지원하는 모든 플랫폼과 완벽한 호환성을 유지하며 다음과 같은 개발 및 운영 환경에서 사용된다:

- Caché 터미널의 명령 줄에서 대화식으로 사용
- Caché 오브젝트 클래스 메소드의 실행 언어
- Caché 실행 프로그램인 Caché ObjectScript 루틴 생성
- Caché SQL 저장 프로시저 및 트리거의 실행 언어
- CSP(Caché Server Page) 응용프로그램의 서버 실행 스크립트 언어

Caché ObjectScript는 Caché의 또 다른 스크립트 언어인 Caché Basic과 완벽하게 호환 가능할 뿐만 아니라 병행 사용이 가능된다.

### 1) 특징

Caché ObjectScript 은 다음과 같은 주요 특징들을 포함하고 있다:

- 문자열 데이터의 작업을 위한 매우 효과적인 내장 함수 제공.
- 메소드, 프로퍼티, 다형성 등을 포함한 완벽한 오브젝트 기능 지원.
- 응용프로그램에서의 워크플로우 제어를 위한 다양한 명령어들.
- 입출력 장치 제어를 위한 일련의 명령어들.
- 로컬, 글로벌 모두에 적용되는 다차원 및 희소 배열 지원
- 효율적인 임베디드 SQL 지원
- 명령어의 실시간 평가 및 실행, 그리고 간접 실행 지원

### 2) 언어 개요

다음은 Caché ObjectScript 의 주요 구성에 대한 설명이다.

Caché ObjectScript 는 특정 예약어를 정의하지 않기 때문에, 변수명과 같은 식별자로서 일반적 단어들을 자유롭게 사용할 수 있다. Caché ObjectScript는 사용자 정의 식별자와 ObjectScript 엘리먼트를 구별하기 위해서 일련의 내장 명령어들과 (내장 함수에 붙는 \$ 와 같은) 특수 문자를 사용한다.

예를 들어, 하나의 변수에 값을 지정하기 위해서는 다음과 같이 **SET** 명령어를 사용한다:

```
SET x=100  
WRITE x
```

또한, (비록 권장 사항은 아니지만) 다음과 같이 명령어 자체를 변수명으로 사용할 수도 있다:

```
SET SET=100  
WRITE SET
```

명령어나 함수명과 같은 Caché ObjectScript 컴포넌트들은 대소문자를 구별하지 않는 반면에 변수나 메소드명은 대소문자를 구별한다.

여백은 거의 모든 곳에서 자유롭게 삽입, 생략 가능하지만 한가지 유의할 점은 루틴 내의 소스에서 명령 라인은 반드시 라인의 두 번째 행에서부터 시작하여야 한다는 것이다. 즉, 모든 명령어는 행의 첫머리에서 들여쓰기로 시작되어야 한다. 유일한 예외는 루틴 **레이블(label)**로서 반드시 첫 번째 행에서부터 시작하여 명령 라인과 구별되어야 한다:

```
MyLabel  
SET x=100  
WRITE x
```

### 3) 명령어, 함수의 호출

ObjectScript 체계는 아래와 같이, 표현식에 대한 명령의 실행도 제공한다:

```
WRITE x
```

즉, 변수 x 에 대해 **WRITE** 명령어를 실행하여 x 값을 출력한다. 여기에서 x 는 표현식이다.

Caché ObjectScript 표현식은 값을 구하기 위해 계산되는 하나 이상의 “토큰”들로 구성된다. 각각의 토큰은 문자, 변수, 하나 이상의 연산자들의 실행 결과, 함수의 결과값, 또는 이러한

것들의 조합일 수 있다. 이러한 Caché ObjectScript 구문은 명령어, 함수, 그리고 연산자 및 표현식등으로 이루어져 있다.

### 명령문과 명령어

Caché ObjectScript 프로그램은 하나 이상의 명령문으로 구성된다. 각 명령문은 프로그램이 실행할 특정 동작을 규정하며, 하나의 명령어와 인수들로 구성된다.

```
SET x="World"  
WRITE "Hello",!,x
```

여기에서 **WRITE** 명령은 인수(들)로서 설정된 값(들)을 현재 사용중인 디플트 출력 디바이스에 출력하는 것이다. 위의 문장들을 Caché 터미널에서 실행하는 경우 WRITE 는 3 개의 인수값들을 Caché 터미널에 출력한다. 즉 문자열인 “Hello”, 라인피드(LF) 및 카트리지리턴(CR)를 실행하는 특수 연산자 “!” 문자, 그리고 로컬변수 x 의 현재 값이다. 인수들은 콤마로 구분한다.

대부분의 Caché ObjectScript 명령어들(그리고 다수의 함수들과 특수 변수들)은 정규 양식과 약어(일반적으로 하나 또는 두 개의 문자로 축약된 명령어)를 사용한 약식 양식으로 동시에 사용 가능하다. 예를 들어, 위의 명령문은 아래와 같이 정의하여도 동일한 결과를 리턴한다:

```
S x="World"  
W "Hello",!,x
```

### 함수

함수는 자주 필요로하는 작업(예를들어, 문자열을 아스키(ASCII) 코드 값으로 전환하는 작업)을 위해 동일 형식으로 반복적으로 사용할 수 있도록 정의된 루틴을 말한다. 함수는 명령문내에서 호출되며 함수를 호출할 때 경우에 따라 필요한 파라미터를 같이 전달함으로써 동일 형식의 루틴이지만 조건과 파라미터값에 따라 약간 변경된 작업을 수행할 수 있도록 정의할 수도 있다. 함수는 작업의 결과를 단일 타입의 값으로 리턴한다. 표현식을 사용할 수 있는 곳에서는 어디서나 함수 사용이 가능하며, 오브젝트 환경에서 호출되는 함수는 메소드라고 정의한다.

Caché 가 제공하는 임베디드 함수들외에 개발자들은 Caché ObjectScript 을 사용하여 자체적으로 사용자 정의 함수인 “프로시저”를 정의, 개발할 수 있다.

## 표현식

표현식은 단일 값을 산출하기 위해 계산되는 토큰들의 집합이다. 예를 들어, 아래의 “Hello”라는 문자열과  $1 + 2$  는 일종의 표현식이다. 마찬가지로,  $x$  와 같은 변수, `$LENGTH()` 와 같은 함수, `$ZVERSION` 과 같은 특수 변수도 표현식이라 하겠다:

```
SET x="Hello"  
WRITE x,!  
WRITE 1+2,!  
WRITE $LENGTH(x),!  
WRITE $ZVERSION
```

## 변수

Caché ObjectScript 에서 변수는 실행중인 값이 저장되는 장소의 이름을 말한다. 변수에는 로컬, 글로벌, 프로세스 전용 글로벌, 배열, 오브젝트 프로퍼티, 특수 변수 등이 있으며 이에 대한 소개는 ”변수” 장에서 자세히 설명하기로 한다.

## 연산자

Caché ObjectScript 는 많은 내장 연산자를 제공하고 있는데, 더하기(“+”), 곱하기(“\*”) 같은 산술 연산, 논리 연산, 패턴 매치 연산 등이 있다. 연산자에 대한 자세한 설명은 “연산자” 장에서 소개하기로 한다.

## 4) ANSI 표준 M 과의 관계

Caché ObjectScript 는 미국표준협회(ANSI)의 표준 M 프로그램 언어를 기반으로 발전된 상위 시스템으로서 M 프로그램을 사용하는 경우, 거의 변경 없이 Caché 에 적용 사용할 수 있다. Caché ObjectScripts 는 ANSI 의 표준 M 기능과 더불어 다음과 같은 기능들이 포함된 다수의 주요 최신 기법들을 추가 제공하고 있다.

- 오브젝트 및 객체지향 프로그래밍을 위한 통합 환경 지원
- { }를 사용한 프로시저와 제어 블록
- 여백 사용 규제의 완화
- 기타 많은 새로운 함수들

## 2. 변수

변수는 값을 저장할 수 있는 위치명이며, Caché ObjectScript에서 변수는 특정 데이터타입을 갖지 않으므로 타입을 선언할 필요가 없다. 변수는 SET 명령을 사용하여 정의되며 값을 설정받는다.

### 1) 변수의 범주

Caché ObjectScript 변수는 각각의 목적과 적용 범위에 따라 다음과 같이 분류한다:

- 로컬 변수
- 프로세스 전용 글로벌
- 글로벌 변수(또는 글로벌)
- 특수 변수(또는 시스템 변수)

#### 로컬 변수

로컬 변수는 현재 실행중인 Caché 프로세스에 속한 변수를 말한다. 로컬 변수는 이것을 생성한 프로세스에서만 사용 가능하며, 프로세스 종료와 함께 소멸된다. 로컬 변수의 SET 또는 KILL은 저널되지 않으며 데이터베이스 트랜잭션 적용을 받지 않는다.

#### ➤ 명명 규약

로컬 변수는 다음과 같은 명명 규약을 따른다:

- 로컬 변수명은 유효한 식별 방법을 사용해야 하며 변수명의 첫 문자는 알파벳이거나 % 문자이어야 한다. % 문자로 시작하는 로컬 변수는 일명 퍼센트 변수라고도 하는데 특별한 사용 범위를 갖기 때문이다. "%Z" 또는 "%z"로 시작하는 변수들만이 사용자 어플리케이션에서 정의하여 사용할 수 있으며, 그 외의 퍼센트 변수들은 Caché 시스템만이 사용할 수 있다. 또한 % 문자는 로컬 변수명의 첫번째 문자로만 사용할 수 있으며, 그 이외의 문자들은 알파벳 또는 숫자로만 이루어져야 한다. 유니코드 시스템의 경우, 아스키(ASCII) 128 이상의 문자도 사용할 수 있다.
- 모든 단어를 변수명으로 사용할 수 있으나, 가능하면 Caché ObjectScript의 명령어나 SQL 예약어는 사용하지 말 것을 권장 한다.
- 로컬 변수는 대소문자를 구분한다. 예를 들어, MYVAR, MyVar, myvar 등은 각각 별개의 로컬 변수로 취급한다.
- 로컬 변수명은 단일 프로세스내에서는 유일한 이름이어야 한다. 즉, 다른 프로세스에서는 동일한 이름의 로컬 변수명을 정의하여 사용할 수가 있다. 또한 프로세스 전용 글로벌 또는 글로벌들도 로컬 변수들과 동일한 이름을 갖을 수 있다. 예를 들어, myvar, ^||myvar, ^myvar는 각각 별개의 변수를 의미한다.

- 로컬 변수명의 길이는 31 개의 문자로 제한된다. 그 이상의 문자로 변수명을 표시할 수는 있지만 31 문자만 유효하게 적용됨으로 처음 31 개의 문자 내에서 유일한 이름으로 정의되어야 한다.
- 로컬 변수들은 첨자로 구성될 수 있는데, 첨자를 사용함으로써 로컬 변수를 배열로 정의할 수 있게 된다. 예를 들어, myvar, myvar(1), myvar(1,2) 등으로 정의함으로써 로컬 변수 myvar 를 배열로 처리할 수 있다.

#### ➤ 타입, 사용법, 사용 범위

Caché 에는 다음과 같이 3 종류의 로컬 변수를 정의할 수 있다:

- 전용(Private) 변수: 프로시저 블록에서 사용되는 모든 변수는 자동적으로 전용 변수로 정의되어 프로세스 블록에서만 액세스 가능하다. 디폴트로, 모든 오브젝트 메소드는 프로시저 블록(클래스 정의에 **ProcedureBlock** 클래스 키워드가 설정됨)으로 정의되며 따라서 메소드내에서 정의되는 모든 변수들을 디폴트로 전용 변수가 된다. 전용 변수에서는 **NEW** 명령어를 사용할 수 없다.
- 공용(Public) 변수: 공용 변수는 변수가 **NEW** 명령어로 다시 정의되거나 프로그램이 **프로시저 블록**에 들어간 경우 이외에는, 현재의 프로세스내의 모든 영역에서 참조 가능한 변수이다.
- % 변수: %로 시작되는 변수는 항상 공용 변수로 취급되어, 프로세스내의 모든 영역에서 참조 가능한 특수 변수이다. "%Z" 또는 "%z" 로 시작하는 변수만이 사용자 어플리케이션 코드에서 정의할 수 있으며, 그 외의 다른 퍼센트 변수들은 시스템만이 사용할 수 있다.

인수 없이 **WRITE** 또는 **ZWRITE** 명령어를 사용하면 현재 프로세스에서 정의된 로컬 변수들의 목록을 볼 수 있다. **\$QSUBSCRIPT** 함수를 사용하여 주어진 로컬 변수의 구성요소(이름 및 첨자)들을 얻을 수 있으며, **\$QLength** 함수를 사용하여 첨자 계층의 깊이를 알 수 있다. 한편 **KILL** 명령을 사용하여 로컬 변수를 삭제한다.

#### **프로세스 전용(Process-private) 글로벌**

프로세스 전용 글로벌은 해당 글로벌을 정의한 프로세스에서만 사용되는 변수로 프로세스가 종료되면 로컬 변수처럼 함께 소멸된다. 다만, 로컬 변수처럼 프로세스 메모리 영역을 사용하는 것이 아니고 Cache 공유 메모리(글로벌 버퍼) 영역을 사용함으로써 대량의 데이터를 처리할 때 유용하게 사용할 수 있다.

대부분의 경우, CacheTemp 데이터베이스의 역할을 대신하지만 CacheTemp 와는 달리 프로세스 종료시 자동적으로 데이터가 소멸된다. 또한, 프로세스 전용 글로벌에 대한 **SET** 또는 **KILL** 은 저널되지 않으며 데이터베이스 트랜잭션 적용을 받지 않는다.

## ➤ 명명규약

프로세스 전용 글로벌은 다음과 같은 형식 중 한가지 형태를 갖게 된다:

```
^||name  
^|^"^|name  
^["^"]name  
^["^",""]name
```

이 네 가지 형식은 모두 동등한 것으로서 동일한 프로세스 전용 글로벌 변수를 참조한다. 첫번째의 (^||name) 형식이 가장 일반적으로 사용되는데, 새로운 코드에 권장하는 형식이다. 나머지 형식들은 이미 사용중인 기존의 코드와 호환되기 위해 제공되는 것이다.

프로세스 전용 글로벌은 다음의 명명 규약을 따른다:

- 프로세스 전용 글로벌은 반드시 유효한 식별자이어야 하며, (두번째 바 이후의) 첫번째 문자는 알파벳이거나 % 문자이어야 한다. % 문자는 이름의 첫 문자로만 사용될 수 있으며 "%Z" 또는 "%z"로 시작하는 변수만 사용자 어플리케이션에서 정의하여 사용될 수 있고 (^||%zmyppg 또는 ^||%Z123과 같이), 그 외의 다른 퍼센트 변수는 Caché 시스템만이 사용할 수 있다.

프로세스 전용 글로벌의 두 번째 이후의 문자는 글자, 숫자, 마침표로 구성할 수 있지만, 마침표 문자는 이름의 첫번째와 마지막 문자로 사용할 수 없다. 유니코드 시스템에서는 아스키(ASCII) 128 이상의 문자도 사용할 수 있다.

- 프로세스 전용 글로벌명은 대소문자를 구별한다.
- 프로세스 전용 글로벌명은 해당 프로세스내에 유일하여야 한다.
- 프로세스 전용 글로벌명은 31 개의 문자로 제한된다. 그 이상으로 변수명을 표시할 수는 있지만 처음 31 문자만이 유효하게 적용되므로 처음 31 개 문자 내에서 유일한 이름으로 정의되어야 한다.
- 프로세스 전용 글로벌은 첨자로 구성될 수 있는데, 첨자로 구성됨으로써 프로세스 전용 글로벌을 배열로 정의할 수도 있다.

## 글로벌

글로벌은 Caché 데이터베이스에 자동 저장되는 특별한 종류의 변수이다. 글로벌은 특정의 네임스페이스에 매핑되어 있으며 확장 참조를 사용하지 않는 한, 글로벌을 정의한 네임스페이스에서만 사용할 수 있다. 글로벌은 네임스페이스내의 모든 프로세스에서 액세스

가능하며 글로벌을 생성시킨 프로세스가 종료하더라도 명시적으로 삭제하지 않는 한 데이터베이스에 계속적으로 남아있게 된다.

글로벌에 대한 **SET** 또는 **KILL**은 저널되며 데이터베이스 트랜잭션의 적용을 받는다.

Caché ObjectScript 프로그램에서 글로벌은 다른 변수들과 동일한 방법으로 사용할 수 있으며, 다만 글로벌임을 나타내기 위하여 글로벌명은 “^” 문자로 시작한다. 다음의 예에서, 변수 mylocal 은 로컬 변수를, ^myglobal 은 글로벌 변수를 의미한다:

```
SET mylocal = "로컬 변수"  
SET ^myglobal = "현재 네임스페이스에 저장되는 글로벌"
```

글로벌은 아래의 명명 규약을 따른다.

- 글로벌은 “^” 문자와 이름으로 구성되며, 현재의 네임스페이스에 정의된 글로벌을 의미한다. 다른 네임스페이스에 있는 글로벌을 액세스하는 경우, ^|namespace|global\_name 형식을 취한다. 예를 들어, ^|"samples"|"Person 은 네임스페이스 “SAMPLES”에 저장된 글로벌 ^Person 을 액세스하는 것이다.
- 글로벌명은 반드시 유효한 식별자이어야 하며, “^” 이후의 첫 문자는 글자이거나 % 문자이어야 한다. "%Z" 나 "%z"로 시작하는 변수만 사용자 어플리케이션에서 정의하여 사용할 수 있으며 그 외의 다른 퍼센트 변수는 Caché 시스템만이 사용할 수 있다. 글로벌의 두 번째 이후의 문자는 글자, 숫자, 마침표로 구성되는데, 마침표는 글로벌명의 첫번째와 마지막 문자에는 사용할 수 없다. 유니코드 시스템에서는 아스키(ASCII) 128 이상의 문자를 사용할 수도 있다.
- 글로벌명은 대소문자를 구별한다.
- 글로벌명은 해당 네임스페이스에서 유일한 이름이어야 한다.
- 글로벌명은 “^”를 제외하고 31 개의 문자로 제한된다. 그 이상으로 변수 명을 표시할 수는 있지만 처음 31 문자만 유효하게 적용되므로 글로벌명은 반드시 처음 31 개 문자 내에서 유일한 이름으로 정의되어야 한다.
- 글로벌은 첨자로 구성될 수 있는데, 첨자로 구성됨으로써 글로벌을 배열로 정의할 수도 있다.

글로벌에서는 선택적으로 “^” 문자 바로 다음에 || 부호나 꺽새괄호를 사용 (예를 들면 ^|"samples"|"myglobal 또는 ^||"|"myglobal) 다른 네임스페이스나 디렉터리를 설정하는 확장 참조 표시를 사용한다. 따라서 이러한 확장 참조 표시를 프로세스 전용 글로벌(예, ^||"myglobal)과 혼동되지 않도록 유의하여야 한다.

**\$ZREFERENCE** 특수 변수를 사용하여 가장 최근에 사용된 글로벌명을 얻을 수 있고, **\$QSUBSCRIPT** 함수를 사용하여 특정 글로벌의 구성요소를 반환 받을 수도 있으며, **\$QLLENGTH** 함수를 사용, 문자 계층의 깊이를 알 수도 있다.

## 배열 변수

배열 변수는 하나 이상의 문자 계층을 가지는 변수를 말한다. 문자는 괄호로 표시되며, 콤마로 구분된다. 다음과 같이 모든 변수는 특수 변수를 제외하고는 배열 변수로 사용될 수 있다:

```
SET a(1) = "로컬 변수 배열"
SET a(1,1,1) = "다른 종류의 로컬 변수 배열"
SET ^||a(1) = "프로세스 전용 글로벌 배열"
SET ^a(1) = "글로벌 배열"
SET obj.a(1) = "다차원 배열 프로퍼티"
```

## 특수 변수(시스템 변수)

Caché ObjectScript 에서는 사용자 어플리케이션에서 사용할 수 있는 시스템 정보들을 리턴하는 다수의 특수 변수들을 제공한다. 모든 특수 변수는 "\$" 문자로 시작되며 사용자는 임의로 특수 변수를 정의할 수 없다. 특수 변수들은 추가적인 매팅 정의없이 모든 네임스페이스에서 사용 가능하다.

특수 변수는 현재 운영 환경의 여러 상황들에 대한 값을 갖고 있으며, 몇몇 변수의 경우 null("") 값으로 초기화되어 있어 <UNDEFINED> 오류를 생성하지 않는다. 특수 변수는 현재의 프로세스에 기반하며 다른 프로세스로부터 참조할 수 없다.

몇몇 특수 변수들에 대하여 사용자가 **SET** 명령을 사용할 수 있지만, 대부분의 경우는 수정할 수 없는 읽기전용 값들이다. 특수 문자 각각에 대한 자세한 정보는 Caché 온라인 문서 Caché ObjectScript Reference 를 참조하기 바란다.

다음은 몇몇 특수 변수들의 예이다:

```
SET starttime = $HOROLOG
HANG 5
WRITE !,$ZDATETIME(starttime(시작 시간))
WRITE !,$ZDATETIME($HOROLOG)
```

특수 변수 **\$HOROLOG** 는 현재 시스템의 날짜와 시간을 시스템 형식(ddddd,nnnnn)으로 리턴하는 변수이며, **\$ZDATETIME**은 사용자가 일반적으로 사용하는 텍스트 형식으로 리턴한다.

또한 \$JOB 은 프로세스 ID 를, \$ZVERSION 은 현재 사용중인 Caché 버전을 리턴한다:

```
WRITE !,"$JOB = ",$JOB // Current process ID  
WRITE !,"$ZVERSION = ",$ZVERSION // Version info
```

### 오브젝트 프로퍼티

오브젝트 프로퍼티는 특정 오브젝트 인스턴스와 결합된 값을 말한다. 엄밀한 의미에서, 오브젝트 프로퍼티는 변수라 할 수 없지만, 문법적으로 변수와 동일한 방법으로 사용할 수 있다.

```
// 주소 오브젝트 생성하기  
SET address = ##class(Sample.Address).%New()  
  
// 오브젝트의 프로퍼티 사용하기  
SET address.City = "Boston"  
WRITE "City: ", address.City,!
```

## 2) 변수 타입 정하기와 전환

Caché ObjectScript 는 변수의 타입을 정의하지 않으며 기본적으로 모두 텍스트 타입이다. (자바스크립트, VB 스크립트, Caché Basic 에서도 동일함). 따라서, 스트링 값을 사용하던 변수에 필요에 따라 숫자 값을 설정할 수도 있다. 그렇지만 내부적으로 Caché 는 문자열, 정수, 숫자, 오브젝트를 각각 다른 타입으로 취급하고 있다. Caché 는 사용중인 문맥을 기반으로 자동적으로 변수의 값을 다른 타입으로 전환시킨다:

```
// 변수 설정하기  
SET a = "This is a string"  
SET b = "3 little pigs"  
SET int = 22  
SET num = 2.2  
SET obj = ##class(Sample.Person).%New()  
  
// 표시하기  
WRITE "a: ",a,!  
WRITE "b: ",b,!  
WRITE "int: ",int,!  
WRITE "num: ",num,!  
WRITE "obj: ",obj,!  
  
// 다른 타입으로 사용하기  
WRITE "+a: ",+a,!  
WRITE "+b: ",+b,!  
WRITE "int _ "" abc"": ",int _ " abc",!  
WRITE "num _ "" abc"": ",num _ " abc",!  
WRITE "+obj: ",+obj,! 
```

### Caché ObjectScript 타입 전환 규칙

From	To	규칙
숫자	오브젝트	허용 안됨
숫자	문자열	숫자는 그 값에 대응하는 숫자 문자들의 스트링으로 전환된다. 예를 들면 22는 “22”가 된다.
오브젝트	숫자	오브젝트 참조는 그 값이 오브젝트 인스턴스 값인 정수로 전환된다.
오브젝트	문자열	오브젝트 참조는 “oref@classname” 형식의 스트링 값으로 전환되는데, 여기서 <i>oref</i> 는 오브젝트 인스턴스의 번호이고 <i>classname</i> 은 오브젝트 타입이다.
문자열	숫자	문자열은 숫자 이외의 문자가 나올 때까지 왼쪽에서 오른쪽으로 구문분석된다. 문자열이 숫자로 시작하는 경우 이 숫자들은 대응하는 수의 값으로 전환되며, 숫자로 시작하지 않고 문자로 시작하면 0으로 전환된다. 예를 들어, “123”은 123으로, “abc123”은 0으로 전환된다.
문자열	오브젝트	허용 안됨

### 오브젝트 값

오브젝트 값은 메모리로 로드된 인스턴스를 의미하며, 모든 오브젝트값은 로컬 변수로 설정된다.

```
SET person = ##class(Sample.Person).%New()
WRITE person,!
```

**참고:** *person* 값은 문자열로 전환된 메모리상의 오브젝트 참조(OREF) 값으로서, 데이터베이스에서 오브젝트를 로드하는데 사용되지는 않는다.

또한 오브젝트 표준 점(dot) 인터페이스를 사용하여 해당 오브젝트의 메소드와 프로퍼티를 참조한다:

```
SET person.Name = "El Vez"
```

`$ISOBJECT` 함수를 사용하여 주어진 변수가 오브젝트 값을 가지고 있는지 알아 볼 수 있다:

```

SET str = "A string"
SET person = ##class(Sample.Person).%New()

WRITE "Is string an object? ", $isObject(str),!
WRITE "Is person an object? ", $isObject(person),!

```

오브젝트 값을 글로벌 변수에 설정할 수 없는데, 이런 경우 런타임 오류가 발생한다.

오브젝트 값을 변수(또는 오브젝트 프로퍼티)에 설정하는 경우, 해당 오브젝트의 내부 참조 수가 증가하게 되는데, 오브젝트 참조 수가 0 이 되면, Caché는 자동적으로 오브젝트를 제거한다. (즉, %OnClose 메소드를 호출하여 오브젝트 인스턴스를 메모리로부터 제거시킨다). 예를 들면,

```

SET person = ##class(Sample.Person).%New() // Person 에 1 개의 참조
SET alias = person // 2 개의 참조

SET person = "" // 1 개의 참조

SET alias = "" // 참조 없음. 오브젝트 없어짐

```

### 3) 변수 선언과 범위

다른 언어와는 달리, Caché ObjectScript에서는 변수를 선언하지 않는다. 변수의 적용 범위는 해당 변수가 프로그램에서 실질적으로 사용될 때 정하여 진다.

Caché ObjectScript에서 변수의 적용 범위를 정하는데는 다음의 두 규칙이 적용된다:

- 프로시저 블록에 근거한 표준 (및 최신) 기법: 새로운 어플리케이션에 권장하는 방법으로서 Caché 스튜디오가 사용하는 기본 방법이다. 프로시저 블록에서 % 변수가 아닌 모든 변수는 전용 변수가 된다.
- 기존 메커니즘: 이미 사용중인 이전 버전과의 호환성을 위해 제공되는 방법으로 프로시저 블록 밖에서 모든 변수는 공용 변수가 된다.

#### #Dim 사용

코드를 작성할 때 **#Dim** 기능을 사용할 수 있는데, **#Dim**은 원하는 변수의 타입에 대한 정보를 제공한다. 이 정보는 Studio Assist 기능에 의해 코드 자동 완성에 사용된다. 또한 문서처럼 코드를 보는데 사용되는 툴에 제공되기도 한다. 다음은 **#Dim**의 사용 예이다:

```

#Dim VariableName AS DataTypeName
#Dim VariableName AS LIST OF DataTypeName
#Dim VariableName AS ARRAY OF DataTypeName

```

여기에서 **VariableName**은 변수명이고 **DataTypeName**은 데이터 타입이다.

참고: #DIM은 정식 변수 선언이 아니며 따라서 이에 해당하는 코드가 생성되지 않는다.

### 3. 연산자

Caché에서는 산술 계산, 논리 연산등 다양한 작업들을 수행하기 위한 연산자들을 지원하고 있는데, 연산자는 변수, 개체인 표현식에 사용되어 결과 값을 산출하는데 사용된다. 이 장에서는 다양한 ObjectScript 연산자 및 사용식에 대해 설명한다.

연산자는 피연산자과 같이 사용되어 의도된 작업을 수행하는 기호 문자를 말한다. 각각의 피연산자는 하나 이상의 표현식이나 표현식 원자로 구성되며, 연산자 및 관련 피연산자는 다음의 형식을 취한다:

[피연산자] 연산자 피연산자

몇몇 연산자들은 하나의 피연산자만을 취하는데 이를 단항(unary) 연산자라 하며, 두 개의 피연산자를 취하는 연산자는 이진(binary) 연산자라 한다.

하나의 연산자와 결합된 피연산자들이 하나의 표현식을 구성하게 되는데, 이 표현식이 피연산자들에 대한 연산자의 계산을 이용하여 결과를 생성하는 것이다. 표현식은 연산자의 타입에 따라 다음과 같이 분류된다.

- 산술 표현식: 산술 연산자를 가지며, 피연산자에 수리적 계산을 적용하여 결과값을 숫자로 생성한다.
- 문자열 표현식: 문자열 연산자를 가지며, 피연산자에 문자열 계산을 적용하여 결과값을 문자열로 생성한다.
- 논리 표현식: 관계 및 논리 연산자를 가지며, 피연산자에 논리적 계산을 적용하여 참(1) 또는 거짓(0)의 논리 결과를 생성한다.

#### ObjectScript 연산자

연산자	실행 작업
.	오브젝트 프로퍼티 또는 메소드 액세스
()	배열 인덱스 또는 함수 호출 인수
+	더하기(이진), 플러스 (단항)

-	빼기(이진), 마이너스 (단항)
*	곱하기
/	나누기
₩	정수 나누기
**	거듭제곱
#	계수(나머지)
-	연결
'	논리적 부정 (NOT)
=	등식, 지정(Assignment)
'=	부등식
>	보다 큼
'>	보다 크지 않음(같거나 작음)
<	보다 작음
'<	보다 작지 않음(크거나 같음)
[	포함
]	그 다음
]]	이후 정렬
&&, &	논리적 AND(&&는 생략 AND 를 표시)
, !	논리적 OR(  는 생략 OR를 표시)

@	간접 명령
?	패턴 매치

## 연산자 우선 순위

ObjectScript에서 연산자 순위는 무조건 왼쪽에서 오른쪽으로 시행된다. 즉, 연산자 타입에 상관없이, 설정된 순서에 따라 진행된다. 이것은 다른 언어에서처럼 특정 연산자가 다른 것보다 우선하는 경우와는 다른 것임으로 유의해야 한다. 그러나 괄호를 사용하여 표현식에서 특정 연산자의 우선 실행을 설정할 수는 있다:

```
WRITE "1 + 2 * 3 = ", 1 + 2 * 3,! // 9 를 반환
WRITE "2 * 3 + 1 = ", 2 * 3 + 1,! // 7 를 반환
WRITE "1 + (2 * 3) = ", 1 + (2 * 3),! // 7 를 반환
WRITE "2 * (3 + 1) = ", 2 * (3 + 1),! // 8 를 반환
```

괄호를 사용하여 계산 순서를 변경할 수 있는데, 괄호로 원하는 표현식(산술 및 비교 연산자)을 묶어 표현식의 작업 순위를 제어 한다:

```
SET TorF = ((4 + 7) > (6 + 6)) // False (0)
WRITE TorF
```

위의 예에서, 괄호 사용으로, 4+7이 먼저 계산되고, 다음으로 6+6이 계산된 후, 11>12가 계산되어 최종적으로 0 (false)을 리턴하는 계산이다. 반면에 아래의 예는,

```
SET Value = (4 + 7 > 6 + 6) // 7
WRITE Value
```

단순히 왼쪽에서 오른쪽으로 진행되어 4+7이 계산되고, 그 결과값 11은 6과 비교 계산되어 결과값 1을 생성한 후, 이 1과 6을 더하여 결과값 7을 리턴하는 것이다.

**참조:** 우선 순위는 결과값의 타입도 결정하게 되는데, 이는 마지막으로 수행되는 연산자의 계산 유형을 따르게 된다. 즉, 위의 첫번째 표현식처럼 마지막 연산이 논리 연산이면 논리값(참 또는 거짓)을, 두 번째처럼 산술 연산이면 최종 결과값을 숫자로 리턴한다.

## 4. 명령어

명령어는 Caché ObjectScript 프로그램에서 코드의 기본 단위로서, 모든 실행작업은 명령어에 의해 제어된다. Caché ObjectScript 명령어는 영어 문장과 비슷한데, 단일 영어 문장이 하나의 완전한 생각을 표현하는 것처럼, Caché ObjectScript 명령어도 단일 명령어가 실행할 하나의 작업 단위를 완전하게 표현한다. 예를 들어, 다음 ‘**WRITE**’ 명령은

```
WRITE "Hello"
```

단어 자체가 의미하듯 한 번의 실행으로 “Hello”라는 단어를 현재 사용중인 디바이스에 출력하는 작업을 완료한다.

**참조:** Caché ObjectScript 명령어는 대소문자를 구별하지 않으며 대부분의 경우, 약어 형식으로도 사용할 수 있다. 예를 들어, “**Write**”, “**writE**”, “**W**”, 및 “**w**”는 모두 WRITE 명령을 수행하는 유효한 형식이다. 각 명령어의 약어 목록은 Caché 온라인 문서 *Caché ObjectScript Reference*의 “Table of Abbreviation”을 참조하기 바란다.

Caché ObjectScript 명령어는 다음과 같이 각자 역할에 따라 분류된다:

- 코드 호출: DO, QUIT, JOB, XECUTE
- 지정: SET, KILL, MERGE, NEW
- 흐름 제어: IF, ELSEIF, ELSE, FOR, WHILE, DO/WHILE
- 입출력: WRITE, READ, I/O 명령들

이 장에서는 명령어들의 인수 및 후조건문등에 대해 알아 보도록 한다.

### 1) 명령어 인수

명령어 다음에 오브젝트나 명령어의 범위를 명시하기 위해 인수들을 사용할 수 있다. 예를 들어, 다음의 SET 명령어는

```
SET x = y * 2
```

“x = y \* 2”라는 하나의 인수를 사용하고 있다. SET 명령은 항상 등식 = 기호를 사용하여야 한다. 또한 명령어가 인수를 가지는 경우, 명령어와 첫 인수 사이에는 반드시 한 칸의 여백을 두어야 한다.

그렇지만 후조건 표현식의 경우, 명령어와 후조건문 사이에는 여백이 없어야 하며, 대신 후조건과 첫번째 인수 사이에는 반드시 한 칸의 여백을 두어야 한다. 다음은 명령어 QUIT 의 사용 예이다:

```
QUIT x+y  
QUIT x + y  
QUIT:x<0  
QUIT:x<0 x+y  
QUIT:x<0 x + y
```

### 복수의 인수(Multiple arguments)

대부분의 명령어들은 복수의 인수 사용을 허용하는데, 인수들 간의 구별은 콤마(,)를 사용한다. 예를 들어, 다음의 **SET** 명령은,

```
SET x=2, y=4, z=6
```

3 개의 인수를 사용하여 3 개의 변수에 값을 설정하는 예이다. 이 경우 3 개의 인수들은 서로 독립적으로 순서대로 실행되기 때문에 각 인수들에 각각의 **SET** 명령어가 주어진 것과 동일한 효과를 가진다.

인수 사이에는 여백이 필요 없지만, 인수들간의 구별을 위해 한 칸 이상의 여백을 사용할 수도 있다 - 여백의 개수는 명령 실행에 영향을 주지 않는다. 또한, 인수들 사이에 줄 바꿈, 탭, 주석 등을 사용할 수도 있다.

### 인수 없는 명령어

인수 없이 실행되는 명령어들도 있고 선택적으로 인수를 사용하는 명령어들도 존재한다. 예를 들어, **HALT** 의 경우는 인수가 없는 명령어이며, **DO** 명령어처럼 인수와 같이 사용할 때와 인수 없이 사용할 때, 약간의 다른 작업을 실행하는 명령어들도 있다.

인수 없는 명령어의 경우, 명령어 이후 여백을 둘 필요는 없다. 그렇지만, 동일 라인에서 둘 이상의 인수 없는 명령어를 사용하는 경우, 명령어와 명령어 사이에는 2 칸 이상의 여백을 두어야 한다:

```
QUIT:x=10 WRITE "not 10 yet"
```

위의 예에서, **QUIT** 는 후조건 표현식이지만 인수 없는 명령어로서, **QUIT** 와 **WRITE** 사이에 2 칸의 여백이 존재한다.

#### ➤ 인수 없는 명령어와 중괄호

그렇지만 중괄호를 사용하여 복수의 명령어들을 블록화 할 때, 인수 없는 명령어라 할지라도, 명령어와 중괄호 사이에는 여백 사용에 대한 제약은 적용되지 않는다.

- 즉, 인수 없는 명령어와 열기 중괄호 ({} 사이에는 공백의 유무에 제약받지 않으며, 탭, 줄 바꾸기도 사용할 수 있다. 이것은 FOR 처럼 인수를 가질 수 있으나 인수가 없는 명령어, 또는 ELSE 처럼 인수를 가질 수 없는 명령어에도 동일하게 적용된다:

```

FOR {
    WRITE !,"Quit out of 1st endless loop"
    QUIT
}
FOR{
    WRITE !,"Quit out of 2nd endless loop"
    QUIT
}
FOR
{
    WRITE !,"Quit out of 3rd endless loop"
    QUIT
}

```

- 또한 닫기 중괄호(})와 인수 없는 명령어 사이에도 여백 규칙이 적용되지 않는데 이는 닫기 중괄호가 구별자처럼 적용되기 때문이다:

```

IF 1=2 {
    WRITE "Math error"
}
ELSE {
    WRITE "Arthmetic OK"
    QUIT}
WRITE !,"Done"

```

## 2) 명령어의 후조건(postconditional) 표현식

후조건은 명령어에 뒤따르는 조건문으로서 앞선 명령(또는 명령 인수들)의 실행 여부를 제어하는데 사용하는 조건 표현식이다.

IF, ELSEIFF, ELSE, FOR, WHILE, 및 DO...WHILE과 같은 흐름 제어 명령어를 제외한 모든 Caché ObjectScript 명령어는 후조건문을 사용할 수 있다.

명령어와 하나 이상의 명령어 인수가 후조건문과 함께 사용되면, 명령어 후조건문이 먼저 계산되고, 그 결과가 참(true)인 경우 명령어가 실행된다. 만약 조건문의 결과가 거짓(false)인 경우, 그 인수는 실행이 안되며 명령어 실행은 그 다음 인수에 대한 계산을 수행한다.

## 후조건 문법

명령어에 후조건문을 덧붙이기 위해서는 다음과 같이 명령어 다음에 콜론(:)을 사용하여 조건문을 추가한다:

```
Command:pc
```

여기에서 Command 는 명령어를, pc 는 조건문을 의미한다.

명령어 후조건문은 다음과 같은 규칙을 따른다:

- 명령어(또는 명령어 인수)와 후조건문 사이에는 여백, 탭, 줄 바꾸기, 주석등이 허용되지 않는다.
- 후조건문 표현식이 괄호로 묶여있지 않는 한, 후조건문 표현식 자체에서의 여백, 탭, 줄 바꾸기, 주석등의 사용도 허용되지 않는다. 괄호를 사용한 경우에도, 명령어(또는 인수)와 콜론, 콜론과 열기 괄호() 사이에는 여백, 탭, 줄 바꾸기, 주석등을 사용할 수 없다.
- 후조건문 다음의 여백 사용은 명령어 다음의 여백 사용 규칙과 동일하다. 즉 후조건문 표현식의 마지막 문자와 첫 인수의 첫 문자와의 사이에는 한 칸의 여백이 있어야 하며, 인수가 없는 명령어에서는 후조건문의 마지막 문자와 같은 줄의 다음 명령어 사이에 2 칸 이상의 여백이 필요하다. 단, 인수가 없는 명령어 다음에 중괄호 닫기 기호가 나오면 여백은 필요 없다(괄호가 사용되는 경우, 괄호 닫기는 후조건문 표현식의 마지막 문자로 간주된다).

## 후조건문(형의) 계산

Caché 에서 후조건문은 그 결과를 참(true), 거짓(false)으로 계산하며 1 과 0 의 값으로 리턴한다. 조건문의 결과가 수치적인 값으로 계산되는 경우에도, 결과값이 0 이면 거짓( false)으로, 0 이 아닌 값은 참(true)으로 리턴한다.

- 0 이 아닌 숫자값은 참(true)으로 계산되며, 산술 연산과 같은 숫자 값 기준을 사용한다. 따라서, 1, "1", 007, 3.5, -.007, 7.0 , "3 little pigs", \$CHAR(49), 0\_1" 등은 참(true)으로 계산된다.
- 0 과 null 및 여백만을 갖는 문자열(" ")을 비롯한 모든 비수치(non-numeric) 문자들은 거짓(false)으로 계산한다. 따라서, 0, -0.0, "A", "-", "\$", "The 3 little pigs", \$CHAR(0), \$CHAR(48), "0\_1" 등은 거짓(false)으로 계산된다.
- 표준 Caché 등가 규칙 적용. 따라서, 0=0, 0="0", "a"=\$CHAR(97), 0=\$CHAR(48) 등은 참(true)으로, 0="" , 0=\$CHAR(0), ("=\$CHAR(32)) 등은 거짓(false)으로 계산된다.

다음은, count 에 대한 후조건문의 결과값에 따라 달라지는 WRITE 명령어의 결과를 보여주는 예이다:

```
SET count = 4
WRITE:count<5 "count is less than 5.",!
SET count = 6
WRITE:count>5 "count is greater than 5.",!
```

### 3) 코드 호출

이 장에서는 다음과 같이 DO 명령어 및 DO 관련 명령어에 대해 설명한다.

- DO
- QUIT
- JOB
- XECUTE

#### DO

ObjectScript 에서는 DO 명령을 사용하여 루틴, 프로시저, 함수, 메소드를 호출한다. Caché ObjectScript 에서는 DO 및 SET 명령이 가장 기본적인 역할을 수행하는데, DO 는 다른 코드를 호출하고, SET 는 변수값을 설정한다. DO 의 기본 문법은 다음과 같다:

```
Do ^CodeToInvoke
```

여기에서 *CodeToInvoke* 는 Caché 시스템 루틴이나 사용자 루틴을 의미하며, 루틴명임을 나타내기 위하여 반드시 탈자 기호 “^” 를 루틴명 앞에 붙여야 한다.

또한 루틴 안에 정의된 특정 프로시저가 시작하는 라인의 레이블(또는 태그)만을 호출할 수도 있는데 이 때 호출하고자 하는 프로시저의 레이블은 다음의 예와 같이 탈자 기호 “^” 바로 앞에 위치한다:

```
SET X = 484
DO INT^CodeToInvoke
WRITE Y
```

이 때, 호출되는 INT 프로시저는 반드시 루틴 *CodeToInvoke* 내에 정의되어 있어야 하며 하나의 완전한 작업을 실행하는 로직을 구성하여야 한다.

클래스의 메소드를 호출하는 경우, DO 는 메소드를 명시하는 표현식 전체를 하나의 인수로 취급한다. 인수 표현식의 모양은 호출되는 메소드가 인스턴스 메소드인지 또는 클래스 메소드인지에 따라 다르게 정의된다. 다음은 클래스 메소드를 호출하는 예이다:

```
DO ##class(PackageName.ClassName).ClassMethodName()
```

여기에서 `ClassName` 은 호출하려는 클래스 메소드명이고, `ClassName` 은 해당 메소드를 가지고 있는 클래스명, 그리고 `PackageName` 은 클래스를 포함하고 있는 패키지명이다.

한편 인스턴스 메소드는 로컬 변수로 정의된 오브젝트를 통해 호출된다:

```
DO InstanceName.InstanceMethodName()
```

여기에서 `InstanceMethodName` 은 호출하려는 인스턴스 메소드명이고 `InstanceName` 은 메소드를 갖고 있는 클래스의 인스턴스명이다.

## QUIT

`QUIT` 명령은 메소드를 포함하여 코드의 한 블록 작업을 종료시킨다. 인수없는 `QUIT` 은 단순히 실행의 종료를 의미하며, 인수가 정의되어 있는 경우, 호출 블록을 종료하면서 해당 인수값을 메소드를 호출한 곳으로 리턴한다.

## JOB

`DO` 명령이 사용자와의 상호작업을 위해 전면(foreground)에서 코드를 실행하는 것이라면, `JOB` 명령어는 사용자와의 상호 작업없이, 즉 현재의 프로세스와 독립적으로 백그라운드로 코드를 실행한다. `JOB` 프로세스는 특별히 명시적으로 정의하지 않는 한, 서버 시스템의 현재 환경 구성값을 디폴트로 상속받는다.

## XECUTE

`XECUTE` 는 하나 이상의 Caché ObjectScript 명령어를 특정 작업의 한 단위로 수행하고자 할 때, 이 명령어들의 블록을 하나의 인수로 취하여 주어진 작업을 수행한다. 즉, 인수는 Caché ObjectScript 명령어들로 이루어진 문자열이며 실질적으로 `XECUTE` 는 여러 명령어들을 단일 작업 단위로 실행하는 것이다:

```
SET x="SET id=ans QUIT:ans="" DO Idcheck"  
SET y="SET acct=num QUIT:acct="" DO Actcheck"  
XECUTE x,y
```

위의 코드는 여러 명령어들을 로컬 변수 `x`, `y` 에 문자열로 설정한 후, `XECUTE` 명령에 `x`, `y` 를 인수로 제공하여 정의한 명령들을 일괄 실행하는 예이다.

#### 4) 지정 명령어

- SET
- KILL
- NEW

##### SET

SET 은 변수에 값을 설정하는 명령어이다. 동시에 하나 이상의 변수에 값을 설정할 수도 있다:

```
SET variable = expression
```

위는 단일 변수에 하나의 값을 설정하는 기본 형식으로, 다음과 같은 절차를 거쳐 값이 할당된다:

- ObjectScript 는 표현식을 계산하고 값을 결정한다. 만약 표현식이 정의되지 않은 변수나, 부정확한 문법 사용등과 같은 오류가 발견되면, 설정은 실패하고 관련 오류가 생성된다.
- 변수는 미리 정의할 필요가 없으며 현재 존재하지 않으면, ObjectScript 에서 자동 생성한다.
- 생성된(또는 존재하는) 변수 값을 표현식의 결과값으로 설정한다.

여러 개의 변수에 각각 다른 값을 설정하는 경우에는 다음과 같이 사용한다:

```
SET variable1 = expression1, variable2 = expression2, variable3 = expression3
```

여러 개의 변수에 동일한 단일 표현식을 설정하는 경우에는 다음과 같이 사용한다:

```
SET (variable1,variable2,variable3)= expression
```

다음은 **Person** 클래스의 프로퍼티 **Gender** 에 “Female” 을 설정하는 예로서, 여기에서, person 은 클래스 Person 의 인스턴스 참조값을 갖는 로컬변수이다:

```
SET person.Gender = "Female"
```

또한 복수의 **Person** 오브젝트의 Gender 프로퍼티에 같은 값을 동시에 설정할 수도 있다:

```
SET (per1.Gender, per2.Gender, per3.Gender) = "Male"
```

여기에서 per1, per2, per3 는 Person 클래스의 서로 다른 인스턴스에 대한 각각의 오브젝트 참조값을 갖는 로컬변수들이다.

SET 를 사용하여 값을 리턴하는 메소드를 호출할 수도 있는데, SET 은 메소드를 명시하는 표현식의 리턴 값을 변수, 글로벌, 또는 프로퍼티에 설정한다. 메소드를 호출하는 표현식은

메소드가 인스턴스 메소드인지 또는 클래스 메소드인지의 여부에 따라 결정된다. 다음은 클래스 메소드를 호출하는 형식이다:

```
SET LocalVariable = ##class(PackageName.ClassName).ClassMethodName()
```

여기에서 LocalVariable 은 반환값을 설정받는 로컬변수이고, ClassMethodName 은 호출하려는 클래스 메소드명이며, ClassName 은 메소드를 갖고 있는 클래스명, PackageName 은 클래스를 포함하는 패키지명이다.

다음은 인스턴스 메소드를 호출하는 형식이다:

```
SET LocalVariable = InstanceName.InstanceMethodName()
```

여기에서 LocalVariable 은 반환값을 설정받는 로컬변수이고, InstanceMethodName 은 호출하려는 인스턴스의 메소드명, InstanceName 은 메소드를 갖고 있는 클래스의 인스턴스명이다. 메소드, 클래스 또는 인스턴스가 반환하는 값을 사용하려면, 반드시 SET 명령을 사용하여 호출하여야 한다.

## KILL

KILL 명령은 사용하였던 변수를 메모리에서 제거하거나, 디스크에서도 삭제할 때 사용하며 기본 형식은 다음과 같다:

```
KILL expression
```

여기에서 expression 은 삭제 대상인 하나 이상의 변수이다.

```
KILL x  
KILL y,z
```

KILL 의 특수 형식으로 “exclusive KILL” 이 제공되는데, 이는 지정된 로컬변수를 제외하고 현재 프로세스에 의해 메모리에 정의되어 있는 모든 로컬변수들을 제거하는 것이다. 괄호를 사용하여 괄호안에 삭제를 원하지 않는 변수들을 설정한다. 다음은 변수 x, y, z 가 정의되어 있다고 가정하고 x 이외의 변수 y, z 를 제거하는 예이다:

```
KILL (x)
```

만약 인수가 주어지지 않으면, KILL 은 현재 프로세스에 의해 정의된 모든 로컬 변수들을 제거한다.

## NEW

NEW 명령어는 변수를 초기화하는데 사용한다. 프로시저를 사용하는 응용프로그램에서 프로그램 전체 또는 주요 프로시저에서만 사용할 변수를 초기화하는데 사용한다.

## 5) 흐름 제어 명령어

프로그램의 로직을 구현하는데 있어서 코드 흐름(Flow)을 제어하는 것이 무엇보다도 중요한 일이다. 이를 위해 Caché ObjectScript 에서는 다음과 같은 흐름 제어 명령어들을 지원한다:

- IF, ELSEIF, ELSE
- FOR
- WHILE, DO/WHILE

### IF, ELSEIF, ELSE

루틴의 흐름을 제어하기 위해서는, 조건에 따라 코드의 일정 부문을 실행하거나 생략, 또는 반복 실행하는 기법이 필요하다. Caché 는 이러한 제어를 위한 기본 방식으로 IF 명령을 사용한다.

IF 는 표현식을 조건으로 취하여 그 표현식의 결과가 ‘참(1)’ 또는 ‘거짓(0)’ 인지를 결정한다. ‘참’ 이면, 표현식 다음에 주어진 코드를 실행하고, ‘거짓’ 이면 주어진 코드를 실행하지 않는다. 일반적으로 조건에 따라 실행하여야 할 코드는 여러 명령어들의 블록으로 이루어지며 중괄호로 사용 코드들을 그룹화 한다:

```
READ "Pick a number: ",x
IF x = 78 {
    WRITE !,"It's 78!"
}
```

후조건문을 사용, 코드 블록을 다음과 같이 간단히 할 수도 있다:

```
WRITE:x=78 !,"It's 78!"
```

IF 구성문은 여러 개의 조건을 계산할 수도 있으며, 조건에 따라 서로 다른 코드 블록을 실행하도록 정의할 수 있다. 단순한 단일 명령어와는 달리, IF 구성문은 하나 이상의 인수, 명령어, 및 코드 블록들로 이루어 진다. IF 구성문은 다음과 같은 구조를 취한다:

```
IF if-condition { code } ELSEIF elseif-condition {code} ELSE {code}
```

ELSEIF 와 ELSE 는 필요에 따라 생략 가능한 선택 사항이며, ELSEIF 는 여러 번 사용될 수도 있다. 만약, 두 개의 조건만이 주어진 경우라면, 다음과 같은 구성된다

```
IF if-condition { code } ELSE {code}
```

다음은 IF 구조문의 일반적 예이다:

```
READ "Enter the number of equal-length sides in the polygon: ",x
IF (x = 1) {
    WRITE !,"It's so far away that it looks like a point!"
} ELSEIF (x = 2) {
    WRITE !,"I think that's a line, not a polygon."
} ELSEIF (x = 3) {
    WRITE !,"It's a triangle!"
} ELSEIF (x = 4) {
    WRITE !,"It's a square!"
} ELSE {
    WRITE !,"That's a lot of sides!"
}
```

## FOR

FOR 구조문은 주어진 값에 따라 동일한 코드 블록을 반복 실행한다. 숫자 또는 문자열 값을 기반으로 FOR 루프(loop)를 정의할 수 있다.

일반적으로, FOR 문은 루프의 초기값으로부터 값을 증가시키거나 감소시키는 제어 변수값을 기반으로, 제어 변수값이 종결값에 이를 때까지 코드 블록을 반복 실행한다. 제어 변수가 종결값(end value)에 도달하면 FOR 루프는 종결되며, 종결값이 없는 경우, 루프는 QUIT 명령이나올 때까지 계속된다. 루프 종결시, 제어 변수는 마지막으로 실행되었을 때의 값을 유지하고 있다.

숫자로 이루어진 제어 변수를 기반으로 한 FOR 루프의 일반적 형식은 다음과 같다:

```
FOR ControlVariable = StartValue:IncrementAmount:EndValue {
    // 코드 블록 컨텐츠
}
```

사용되는 모든 값은 플러스, 마이너스 값을 가질 수 있으며, 여백은 허용되나 등식과 콜론 주변에서는 사용하지 않아도 된다. FOR 문의 코드 블록은 변수에 주어진 각각의 값에 대하여 반복 실행된다.

다음의 예는 주어진 코드 블록을 5 번 반복 실행하는 FOR 문이다:

```
WRITE "The first five multiples of 3 are:",!
FOR multiple = 3:3:15 {
    WRITE multiple,!
}
```

또한 변수를 사용하여 종결 값을 결정할 수도 있다. 다음은 변수값으로 루프의 반복 횟수를 지정하는 예이다:

```
SET howmany = 4
WRITE "The first ",howmany," multiples of 3 are "
FOR multiple = 1:1:howmany {
    WRITE (multiple*3)," "
    IF multiple = (howmany - 1) {
        WRITE "and "
    }
    IF multiple = howmany {
        WRITE "and that's it!"
    }
}
QUIT
```

FOR 의 제어 변수들은 특정 값들의 목록으로 주어질 수가 있는데, 이런 경우, 코드 블록은 주어진 목록의 각 항목에 대해 반복 실행 한다:

```
FOR b = "John", "Paul", "George", "Ringo" {
    WRITE !, "Was ", b, " the leader? "
    READ choice
}
```

## WHILE, DO/WHILE

WHILE 과 DO/WHILE 은 서로 유사한 흐름 제어 명령어로서 주어진 조건을 언제 계산하느냐에 따라 차이가 있는데, WHILE 은 코드 블록을 실행하기 전에 먼저 조건을 계산하는 반면에, DO/WHILE 은 코드 블록을 실행한 후 조건을 계산한다. FOR 와 마찬가지로 코드 블록내에 QUIT 명령으로 루프를 종료시킨다.

두 명령어의 일반적 형식은 다음과 같다:

```
DO {code} WHILE condition
    WHILE condition {code}
```

다음은 주어진 값까지의 Fibonacci 수열을 두 번 계산하는 예로서, 한 번은 DO/WHILE 를, 그 다음은 WHILE 을 사용하는 예이다:

```
fibonacci() PUBLIC { // Fibonacci 수열 (배열) 생성하기
    READ !, "어디 값까지(로) Fibonacci 수열(배열) 생성하나요? ", upto
    SET t1 = 1, t2 = 1, fib = 1
    WRITE !
    DO {
        WRITE fib, " " set fib = t1 + t2, t1 = t2, t2 = fib
    }
```

```
WHILE ( fib '> upto )  
  
    SET t1 = 1, t2 = 1, fib = 1  
    WRITE !  
    WHILE ( fib '> upto ) {  
        WRITE fib," "  
        SET fib = t1 + t2, t1 = t2, t2 = fib  
    }  
}
```

**WHILE**, **DO/WHILE**, **FOR**의 차이점은 어디에서 제어 표현식의 값을 테스트 하느냐에 달려 있는데, **WHILE**은 루프 시행 전, **DO/WHILE**은 루프 시행 후, 그리고 **FOR**는 루프상 어느 곳에서도 할 수 있다는 점에 있다. 코드 블록이 두 부분으로 되어있고 두 번째 부분의 실행이 표현식 계산 결과에 의존한다면 **FOR**를 사용하는 것이 가장 적당하며, 그 외의 경우에는, 표현식 계산을 코드 블록 보다 빨리 하는지, 그 후에 하는지에 따라 선택할 수 있다.

## 6) 입출력 명령어

ObjectScript 입출력 명령어는 Caché 외부와의 데이터 교환을 위한 다음과 같은 명령어들을 제공한다:

- **WRITE**
- **READ**
- **I/O commands**

### WRITE

**WRITE** 명령은 데이터를 출력하는 명령어로서 인수 유무에 관계 없이 사용할 수 있다.

적절한 표현식을 인수로 갖는 경우, **WRITE** 는 표현식의 결과값을 출력한다. 반면에 제공되지 않는 경우에는 현재 프로세스의 모든 로컬 변수의 이름과 값을 출력하는데, 글로벌 참조, 프로퍼티, 코드의 결과등은 포함하지 않는다.

예를 들어, 다음은 현재의 출력 디바이스에 문자열 “Hello” 을 출력하는 예이다:

```
WRITE "Hello"
```

다음은 하나 이상의 변수에 대한 예이다:

```
SET x = 4
SET y = 3
WRITE x,! // 4
WRITE y,! // 3
WRITE x,y,! // 43
WRITE "x = ",x,!,"y = ",y // output on multiple lines
```

변수, 글로벌, 프로퍼티 값을 출력하는 경우에도 **WRITE** 명령을 사용한다:

```
SET x = 2
WRITE "x = ",x,!
SET ^x = 4
WRITE "^x = ",^x,!
SET per = ##class(Sample.Person).%OpenId(5)
WRITE "per.Name = ",per.Name,!
KILL x, ^x, per
```

또한 메소드나 함수의 결과를 직접 출력할 수도 있다:

```
WRITE instance.method()
```

여기에서 **instance** 는 메모리에 로드된 오브젝트를 참조하는 로컬 변수이고, **method** 는 참조된 오브젝트의 메소드이다. **WRITE** 에 의한 출력값은 이 메소드의 리턴값이다. 예를 들어, Caché 의 **Sample.Person** 클래스에 **NinetyNine** 이라는 메소드가 정의되어 있어 99 를 리턴한다면, 아래의 예에서, **MyPerson** 은 클래스 **Sample.Person** 의 인스턴스 참조값이고, **WRITE** 명령은 메소드 **NinetyNine()** 의 리턴값 ‘99’ 를 출력한다:

```
SET MyPerson = ##class(Sample.Person).%OpenId(3)
WRITE MyPerson.NinetyNine()
```

값을 리턴하는 루틴이나 다른 ObjectScript 코드를 호출하는 예는 다음과 같다:

```
WRITE $$label^routine()
```

여기에서 **label** 은 루틴에서 정의된 레이블이고, **routine** 은 호출된 루틴명으로서, 리턴값을 갖는 루틴이나 루틴 레이블을 호출할 때에는 시작에 “\$\$” 를, 끝에는 “()” 를 반드시 사용하여야 한다.

다음은 “house” 라는 루틴에 레이블 “Fun” 으로 정의된 서브루틴 코드이며,

```
Fun()
  SET place = "Funhouse"
  QUIT place
```

이를 호출하여 그 리턴값을 출력하는 예이다:

```
USER>WRITE $$Fun^house()
Funhouse
```

WRITE로 서브루틴 호출 시, 서브루틴은 반드시 QUIT을 사용하여 값을 리턴하여야 한다. 만약 그렇지 않은 경우, 오류가 발생한다.

## READ

READ 명령어는 현재의 입력 장치를 통해 사용자가 작성한 데이터를 받아 저장한다. READ 명령은 다음과 같은 인수들을 가질 수 있다:

```
Read format, string, variable
```

여기에서 format은 사용자 입력 영역을 제어하고, string은 입력 프롬프트 앞에 출력되는 문자열이며, variable은 입력 데이터를 저장한다.

다음의 포맷 코드를 사용하여 사용자의 입력 영역을 제어한다:

포맷 코드	효과
!	새로운 줄 시작
#	새로운 페이지 시작. 터미널에서는 현재 스크린의 출력 내용을 모두 지우고 새로운 스크린의 처음에 입력 프롬프트를 위치시킨다.
?n	n(N)은 양의 정수로서 터미널 스크린의 현재 라인의 n 번째 컬럼 위치에 입력 프롬프트를 위치시키다.

## OPEN, USE, CLOSE 명령어

OPEN 명령어는 주어진 디바이스(파일, 프린터, COM, ...)를 통한 입출력을 가능하게 디바이스를 초기화하는 명령이며, USE 명령은 OPEN 명령으로 사용 가능하게 된 디바이스를 사용하기 위해 지정하며, CLOSE 명령은 OPEN 된 디바이스를 더 이상 사용하지 못하게 제거한다.

## 7) 기타 명령어

- 트랜잭션 관련 명령어: **LOCK**, **TSTART**, **TCOMMIT**, **ROLLBACK**
- 디버깅 관련 명령어: **BREAK**, **GOTO**
- **HANG** 명령어: 실행을 일정 시간 중단시킴

기타 명령어들에 대한 자세한 정보는 Caché 온라인 문서 Using Caché ObjectScript 의 관련 부분을 참조하기 바란다.

## 5. 트랜잭션 처리

트랜잭션은 업무의 로직 단위로서, 데이터베이스의 논리적 무결성을 유지하는데 도움을 준다.

예를 들어, 돈을 계좌 이체하는 경우, 은행은 돈을 한 테이블에서 인출해서 다른 테이블로 옮기는 과정이 필요한데, 관련된 두 테이블의 갱신을 단일 비지니스 로직(트랜잭션)으로 정의함으로써, 한쪽 테이블에만 갱신이 발생하는 오류를 방지할 수 있다. 즉, 두 테이블 모두를 갱신하여야만 정상적인 완료가 되는 단일 작업(계좌 이체)의 관련 데이터들의 일관성을 보장 받게된다.

응용프로그램에서, 단일 SQL **INSERT**, **UPDATE**, **DELETE**, 또는 하나의 글로벌 **SET**, **KILL** 명령은 트랜잭션을 구성의 의미를 갖지 않는다. 트랜잭션 처리 명령들은, 일련의 작업들을 묶어 단일한 트랜잭션을 형성할 수 있도록 구성한다. 우선 트랜잭션의 시작을 알리는 명령(**TSTART**)을 트랜잭션에 포함되는 작업들을 시작하기전에 설정한 다음, 트랜잭션을 이루는 일련의 작업들을 수행한 후, 트랜잭션의 종료를 의미하는 명령어(**TCOMMIT**)를 설정한다.

만약 프로그램 오류나 시스템 장애로 인하여 정의된 트랜잭션이 정상적으로 완료되지 않으면, 데이터베이스는 해당 트랜잭션 이전 상태로 룰백(**ROLLBACK**)된다. 어플리케이션 개발자들은 트랜잭션을 정의할 때 반드시 룰백 상황을 적용하여야 한다. Caché 시스템도 시스템 장애시와 같은 프로그램의 정상 종료를 방해하는 상황에서, 진행중이던 미완료 트랜잭션의 자동 룰백을 지원하고 있다.

### 1) 응용프로그램에서의 트랜잭션 관리

Caché에서는 다음 두 가지중 하나의 방법으로 트랜잭션을 정의한다:

- 매크로 소스 루틴에서의 SQL 문장
- Caché ObjectScript 명령어

두 가지 방법 모두, SQL **INSERT**, **UPDATE**, **DELETE** 문 또는 Caché ObjectScript 의 **SET**, **KILL** 명령어와 함께 트랜잭션을 구성하는 데이터베이스의 변경이 실행되었는지 여부와는 관계없이, 작동한다.

## 트랜잭션 명령어

Caché 는 ANSI SQL 연산 COMMIT WORK, ROLLBACK WORK 뿐만 아니라 Caché 확장 %BEGTRANS, %INTRANS 도 지원한다.

다음은 SQL 및 Caché ObjectScript 명령어 테이블이다.

Caché 트랜잭션 명령어

SQL 명령어	ObjectScript 명령어	정의
%BEGTRANS	TSTART	트랜잭션의 시작을 나타냄
%INTRANS	\$TLEVEL	트랜잭션이 현재 진행 중인지 체크 <ul style="list-style-type: none"><li>• &lt;0 : %INTRANS에서 사용. 저널은 안 되지만 트랜잭션중. \$TLEVEL 사용할 수 없음.</li><li>• 0 : 트랜잭션 중이 아님.</li><li>• &gt;0 : 트랜잭션중.</li></ul>
COMMIT WORK	TCOMMIT	트랜잭션의 성공적인 종료를 나타냄
ROLLBACK WORK	TROLLBACK	트랜잭션이 실패하여 트랜잭션 시작이후 실행된 모든 데이터베이스 변경은 룰백 또는 취소되어야 함.

**참고:** Caché 는 글로벌 변수에 대한 SET, KILL 은 저널 트랜잭션 이벤트로 간주하여, 트랜잭션 룰백 시, 실행을 되돌린다. 그렇지만, 로컬 변수나 프로세스 전용 글로벌 변수에 대한 SET, KILL 은 저널 트랜잭션 이벤트로 취급하지 않기 때문에, 트랜잭션 룰백이 적용되지 않는다. 트랜잭션 내에서 수행된 SET, KILL 에 의한 변경된 글로벌 데이터를 트랜잭션이 완료되기 이전에 다른 프로세스에 의해 액세스되는 것을 방지하기 위해서는 LOCK 명령을 트랜잭션과 같이 사용하여야 한다.

## 트랜잭션에서 LOCK 명령어 사용하기

동시에 다수의 사용자에 의해 액세스될 수 있는 가능성이 있는 글로벌을 취급할 때에는, 반드시 해당 글로벌에 대한 LOCK 명령을 적용, 데이터베이스의 무결성을 보장하여야 한다.

다음은 글로벌 ^A에 대한 잠금(lock)을 적용하는 예이다:

```
LOCK ^A
```

단, 이 때, 이미 잠금(lock) 되어져 있는 모든 아이템들은 ^A를 LOCK하기 전에 모두 잠금해제(unlock) 된다. 즉, LOCK 명령은 제공된 인수만을 잡고 이미 잠금되어 있는 아이템들에 대해서는 잠금해제한다. 만약, 인수없이 LOCK 명령을 실행하면 잠금된 모든 항목들이 해제된다.

다음은 글로벌 ^A만을 잠금해제하는 예이다: 즉, 잠금되어 있는

```
LOCK -^A
```

즉, 잠금되어 있는 다른 아이템에는 영향을 주지 않는다. 원칙적으로 잠금 해제는 LOCK 명령을 실행한 프로세스나 사용자에 의해서만 가능하지만, 시스템 관리자는 LOCKTAB 프로그램을 사용하여 어떤 항목이라도 잠금 해제할 수 있다.

만약, 트랜잭션이 완료되기 이전에 잠금해제를 실행하는 경우, 이 잠금해제의 실행은 트랜잭션의 완료(커밋 또는 룰백)후까지 지연되며 잠금 대상 글로벌의 변경 데이터도 액세스할 수 없다.

다음은 트랜잭션내에서의 LOCK 명령 사용의 일반적인 예로서, 트랜잭션에서의 잠금 및 해제 순서를 설명한다. 이 예제는 프로세스가 한 번 실행될 때마다 id 값이 증가하여 ^A(0)에 마지막으로 실행된 프로세스 정보를 기록하는 코드이다:

```
&sql(%BEGTRANS)
LOCK ^A(0) SET id=^A(0)+1,^A(0)=id
LOCK ^A(id) SET ^A(id)=name // processing continues ...
LOCK
&sql(COMMIT WORK)
```

먼저, 트랜잭션 시작전의 ^A(0)의 값을 4라고 가정하자. 트랜잭션 시작 후, 첫번째 라인에서 로컬변수 id는 5가 되고, 이 값은 잠금된 ^A(0)에 설정된다. 이 때, 위의 코드를 호출한 다른 프로세스(두번째)가 있다면, 이 프로세스(두번째)는 id=6의 값을 얻게 된다(두번째 LOCK 명령이 이를 허용). 반면에 현재 프로세스(첫번째)는 트랜잭션을 완료할 때까지 id=5를 계속 유지하게 된다. 이런 순서는 트랜잭션이 정상적으로 종료(커밋)된다면 아무 문제없이 정확한 데이터를 얻을 수 있다. 그렇지만 만약 오류가 발생하여 첫번째 프로세스의 트랜잭션이 룰백되면, ^A(0)값은 4로 재설정되고, 따라서 룰백후 이 트랜잭션을 실행하는 프로세스(세번째)는 id=5의 값을 다시 취하게 된다. 즉, 두번째와 세번째 프로세스가 동일한 값의 데이터를 액세스하여 세번째 프로세스 정보가 두번째 프로세스 정보로서 저장되는 것이다. 이는 프로그램의 논리적 무결성을 위반하는 것이다.

이러한 일이 발생하는 것을 방지하기 위해 Caché 에서는 모든 잠금해제(Unlock) 작업을 트랜잭션이 커밋되거나 롤백될 때까지 연기하는 것이다. 위의 예에서, 두 번째 LOCK 이후 LOCKTAB 을 이용하여 LOCK 테이블을 확인하면,  $^A(id)$  는 잠금되어 있지만  $^A(0)$  는 그렇지 않음을 알 수 있다. 그럼에도 불구하고 현재 트랜잭션이 종료될 때까지는 다른 프로세스가  $^A(0)$ 에 대하여 LOCK 을 적용할 수는 없다.

만약 단일 트랜잭션이 상대적으로 오랜 시간을 요구하며, 비즈니스가 논리적으로 두 개의 트랜잭션으로 분리 가능하다면, 다음과 같이 두 개의 트랜잭션으로 분리하여 프로세스들간 LOCK 대기 시간을 줄여주는 것도 하나의 방법이 될 수 있다:

```
##sql(%BEGTRANS)
LOCK ^A(0) SET id=^A(0)+1,^(0)=id
##sql(COMMIT WORK)

##sql(%BEGTRANS)
LOCK ^A(id) SET ^A(id)=name // processing continues ...
##sql(COMMIT WORK)
LOCK
```

그렇지만, 어떠한 경우에도 논리적 무결성은 항상 보장되어야 함을 명심하여야 한다.

Lock 테이블이 차게 되면, 다음과 같은 메시지를 cconsole.log 파일에 보낸다:

```
LOCK TABLE FULL
```

Lock 테이블 사이즈는 일정하게 정의되어 있기 때문에 동시에 이 사이즈를 넘어서는 아이템들에 잠금이 적용되면 이 메시지가 발생된다. 이것은 교착상태(deadlock)가 아니면 응용프로그램 자체의 문제는 아니지만 동시에 너무 많은 LOCK 명령이 실행되는 것은 지양하는 것이 좋다. 만약, 어플리케이션 특성상 LOCK 명령 빈도를 줄일 수 없다면 Caché 시스템 포탈을 통해 Lock 테이블 사이즈를 증가시켜야 한다. 교착상태(deadlock)의 경우는, 프로세스간 필요한 리소스(글로벌)를 서로 잠금하고 해제하지 않아서 서로 무한정 대기 상태에 빠지게 되는 상태로서, 이것은 프로그램 오류로 간주된다.

### 응용프로그램에서의 트랜잭션 롤백

트랜잭션 작업 중 오류가 발생하면 다음의 3 가지 방법으로 롤백시킬 수 있다:

- SQL 롤백 명령어 **RollBack WORK** 실행
- Caché ObjectScript 롤백 명령어 **TROLLBACK** 실행
- 오류 처리 루틴 **%ETN** 사용

## ➤ SQL 또는 Caché ObjectScript 를백 명령 실행

트랜잭션이 실패했을 경우, 두 가지 타입의 를백 명령어를 사용하여, 완료되지 못한 트랜잭션을 자동적으로 를백시킬 수 있다:

- 매크로 소스 루틴에서 **##sql(ROLLBACK WORK)** 사용
- (매크로 또는 중간 소스 코드에서) Caché ObjectScript 의 TROLLBACK 명령어 사용

루백 명령은 다음과 같이 반드시 에러 트랩(error trap)과 같이 실행되어야 한다:

```
ROU      ##sql(%BEGTRANS) SET $ZT="ERROR"  
        SET ^ZGLO(1)=100  
        SET ^ZGLO=error  
        SET ^ZGLO(1,1)=200  
        ##sql(COMMIT WORK) WRITE !,"Transaction Committed" QUIT  
ERROR    ##sql(ROLLBACK WORK)  
        WRITE !,"Transaction failed." QUIT
```

여기에서, 시스템 변수 \$ZT에 설정된 값은 오류 발생시 자동적으로 실행되는 서브루틴으로 이 서브루틴에서 원하는 방법으로 오류를 처리한다. 위의 예에서, 트랜잭션이 완료 되기 전 오류가 발생하면 서브루틴 “ERROR” 가 실행되어 **##sql(ROLLBACK WORK)** 을 수행하고 사용자가 정의한 오류 메시지를 출력한 후 종료한다. 오류가 발생하지 않고 트랜잭션이 성공적으로 완료되면 **##sql(COMMIT WORK)** 을 수행하고 사용자가 정의한 성공 메시지를 출력한 후 바로 종료한다.

## ➤ %ETN 호출

트랜잭션 를백 명령으로 프로그램에서 트랜잭션 를백을 처리하지 않으면, 시스템 에러 트랩 유ти리티 **%ETN** 은 완료되지 않은 트랜잭션을 감지하고 해당 트랜잭션을 커밋 또는 를백할 건지 여부를 사용자에게 물어본다. 트랜잭션 미완료는 데이터의 논리적 무결성을 저해하는 원인이기 때문에, 트랜잭션 를백은 반드시 비즈니스 로직과 연관하여 관련 어플리케이션 코드에서 이루어져야 한다.

완료되지 않은 트랜잭션이 발견되면 %ETN 은 다음과 같은 를백 프롬프트를 내보낸다:

```
You have an open transaction.  
Do you want to perform a Commit or Rollback?  
Rollback =>
```

만약 10 초 내에 응답을 주지 않으면, 디풀트로 시스템은 를백을 수행한다. 백그라운드 프로세스 또는 어플리케이션 모드 프로세스의 경우에는 메시지 없이 를백하게 된다.

## 응용프로그램상 트랜잭션 진행의 예

다음은 Caché 루틴에서 ObjectScript 명령어와 SQL 코드를 같이 사용하여 트랜잭션(계좌간 자금이체)을 처리하는 일반적인 코드 예제이다:

```
Transfer(from,to,amount) // Transfer funds from one account to another
{
    TSTART
    &SQL(UPDATE A.Account
        SET A.Account.Balance = A.Account.Balance - :amount
        WHERE A.Account.AccountNum = :from)
    IF SQLCODE TROLLBACK QUIT "Cannot withdraw, SQLCODE = " _SQLCODE
    &SQL(UPDATE A.Account
        SET A.Account.Balance = A.Account.Balance + :amount
        WHERE A.Account.AccountNum = :to)
    IF SQLCODE TROLLBACK QUIT "Cannot deposit, SQLCODE = " _SQLCODE
    TCOMMIT
    QUIT "Transfer succeeded"
}
```

## 2) 자동 트랜잭션 롤백

트랜잭션 롤백은 다음의 경우, 시스템 레벨에서 자동적으로 수행된다:

- Caché 시작 때, 이전 서비스와 관련된 복구가 필요한 경우
- HALT
- RESJOB

### Caché 시작시 롤백

Caché 가 시작할 때, 완료되지 않은 트랜잭션이 존재하는 경우, 이를 모두 롤백한(된)다.

### Caché 정지시 롤백

트랜잭션 진행 중일때 HALT 를 사용 프로세스를 종료하는 경우, 시스템은 완료되지 않은 트랜잭션의 커밋 또는 롤백 수행여부를 묻는 프롬프트를 내보낸다. 백그라운드 또는 어플리케이션 모드 프로세스의 경우, 프롬프트없이 롤백을 수행한다.

### RESJOB 작업중 롤백

#### ➤ 백그라운드 또는 어플리케이션 모드 프로세스

트랜잭션 진행중인 백그라운드 프로세스에 대하여 RESJOB 명령을 실행하면, 시스템은 자동으로 롤백을 수행한다.

#### ➤ 프로그래머 모드 사용자 프로세스

프로그래머 모드 사용자 프로세스에서 RESJOB 명령을 실행하면, 시스템은 트랜잭션 커밋할건지 또는 롤백을 원하는지를 물어본다.

#### JOURNAL 유ти리티의 저널 복구 옵션

JOURNAL 유ти리티에서 Restore Journal 옵션을 선택하면, 저널 파일은 복구되고 모든 미완료 트랜잭션은 롤백된다.

### 3) 시스템 레벨에서의 트랜잭션 이슈

이 장에서는 트랜잭션과 관련된 시스템 수준에서의 문제점들에 대하여 설명하고자 한다.

#### 트랜잭션과 백업 및 저널

Caché 의 각 인스턴스는 일정기간 동안의 저널을 보관하게 되는데, 저널은 시스템의 마지막 백업 이후 데이터베이스에서 변경된 데이터들을 시간 별로 기록한 일련의 파일들을 말한다. Caché 트랜잭션은 데이터의 논리적 무결성을 유지하기 위해 저널을 기반으로 실행된다.

저널은 SET/KILL 명령과 함께 변경되는 데이터들을 기록하도록 설정된 데이터베이스에 대하여 적용되는데, 트랜잭션내에서는 이러한 설정과는 상관없이 SET, KILL 명령에 의해 변경된 모든 데이터가 저널에 자동 기록된다.

**백업** 또한 트랜잭션 중에 실행할 수는 있는데, 이런 경우, 생성된 백업 파일에 트랜잭션의 일부분만이 포함될 수도 있다. 이런 백업 파일을 복구하여야 하는 경우에는 우선 백업 파일을 복구한 후, 저널 파일도 적용시켜 트랜잭션을 완료하여야 한다. 저널 파일은 마지막 백업 이후부터 재난 발생 시점까지의 갱신된 모든 데이터를 복구하게 된다. 저널 파일 적용은 미완료된 트랜잭션을 완료시키고 커밋 안 된 트랜잭션을 롤백하여 데이터베이스의 트랜잭션 무결성을 유지하는데 필요하다. 데이터베이스 무결성에 대한 보다 자세한 정보는 온라인 문서 “Caché Data Integrity Guide”를 참조하기 바란다.

## 제 4 장 Caché 오브젝트 모델

### 1. 오브젝트 참조- OREF, OID, ID

- 1) OREF(Object Reference)
- 2) OID(Object Identifier)
- 3) ID(Identifier)
- 4) OID 와 ID 값
- 5) OREF 와 참조 카운트

### 2. 클래스 유형

- 1) 비상주 (transient) 오브젝트 클래스
- 2) 지속 (persistent) 오브젝트 클래스
- 3) 시리얼(serial) 오브젝트 클래스
- 4) 메모리 와 디스크에서의 임베디드 오브젝트
- 5) 데이터 타입 클래스

### 3. 상속

- 1) 다중 상속

### 4. 클래스 컴파일

Caché 오브젝트 모델에서는 **persistent** 오브젝트(데이터베이스)와 **transient** 오브젝트(저장 안 된 것)의 기능과 특징을 설명한다. (Caché 오브젝트 모델은 다음과 같은 기능적인 특징을 가지고 있다.)

- **지속성(Persistence):** 오브젝트는 Caché 데이터베이스 또는 외부 데이터베이스에 저장할 수 있으며 정의된 오브젝트는 동시에 관계형 테이블로도 표현되기 때문에 표준 SQL 을 사용하여 쿼리가 가능하다.

오브젝트 모델은 다양한 모델링을 필요로 하는 데이터베이스 어플리케이션에 필요한 모든 정보들(인덱스, 저장 구조, 무결성 제약조건)을 완벽하게 정의할 수 있을 만큼 풍부한 모델링 환경이다.

- **프로퍼티:** 오브젝트는 단일 텍스트 또는 오브젝트 참조를 값으로 하는 프로퍼티를 가진다. 오브젝트 참조값 프로퍼티는 외부 오브젝트나 임베디드 오브젝트를 참조한다. 그 외에 다양한 관계, 컬렉션, 스트림 프로퍼티등이 존재한다.
- **사용자 정의 데이터 타입:** 오브젝트는 어플리케이션에 필요한 특정 타입을 데이터 타입으로 정의할 수 있는데 이를 사용자 정의 데이터타입이라 한다.
- **메소드:** 오브젝트는 오브젝트 자체와 연관된 기능을 정의하는 메소드를 가진다.
- **다형성(Polymorphism):** 오브젝트는 특정 작업(조건)별로 특화된 실행코드를 제공함으로써 어플리케이션으로 하여금 새로운 환경에 쉽게 적응할 수 있도록 한다. 어플리케이션 실행시 특정 오브젝트 유형에 대응하는 작업을 자동적으로 수행한다.
- **상속(Inheritance):** 어플리케이션은 기존 클래스에 필요에 따라 새로운 컴포넌트들을 추가함으로써 기존 코드를 다시 사용할 수 있게 한다.

개별 클래스는 사람, 기록, 계좌 같은 실체를 디자인하고, 이렇게 디자인된 각 클래스들은 데이터(프로퍼티), 작업(메소드) 및 기타 추가적인 기능들을 정의한다. 오브젝트는 클래스의 특정한 인스턴스로서, 디스크, 메모리 또는 클라이언트 어플리케이션에 존재한다.

## 1. 오브젝트 참조- OREF, OID, ID

Caché에서, 오브젝트는 디스크 또는 메모리에 존재하게 되는데 디스크상에 존재하는 오브젝트는 데이터베이스에 저장되어 있는 것을 의미하며, 메모리에 존재하는 오브젝트는 데이터베이스에서 로드되어 현재 사용자가 액세스할 수 있는 오브젝트를 말한다. 오브젝트가 현재 디스크에 저장되어 있느냐 메모리에 로드되어 있느냐의 여부에 따라 오브젝트를 참조하는 방법이 달라진다:

### 1) OREF(Object Reference)

오브젝트 참조로서, 메모리에 로드되어 있는 오브젝트의 특정 인스턴트를 참조하는 값이다. 동일 오브젝트라 하더라도, 해당 오브젝트가 메모리로 로드될 때마다 시스템에 의해 다른 OREF 값이 설정된다.

## 2) OID(Object Identifier)

오브젝트 식별자. 데이터베이스에 저장된 특정 오브젝트의 인스턴스를 식별하는 오브젝트별 고유 값이다. 즉, 디스크상의 오브젝트 ID 를 의미한다. OID 는 데이터베이스에 저장된 persistent 오브젝트의 유일한 식별 값이다. 오브젝트에 OID 값이 한번 설정되면, 그 값은 변경되지 않는다.

## 3) ID(Identifier)

오브젝트 식별자로서, 주어진 범위 안에서 특정 인스턴스를 식별하는 유일한 값이다. ID 는 디스크상에서의 오브젝트 식별자로서 클래스 정보는 포함하지 않는다.

persistent 오브젝트의 OID 를 사용하여 디스크상의 오브젝트를 검색하여 메모리로 로드하게 되며, 또한 만약 persistent 오브젝트의 범위를 알면, ID 를 사용, 오브젝트를 찾아낸 후, 메모리로 로드할 수 있다. 일단 메모리로 로드되면, 시스템은 이 오브젝트 인스턴스에 OREF 값을 설정하여 어플리케이션에서 해당 오브젝트를 액세스할 수 있도록 한다. persistent 오브젝트가 데이터베이스에 저장되어 있을 때는, 이 오브젝트의 모든 참조 속성들은 OID 의 값으로 저장된다. OID 의 값으로 정의되지 않은 오브젝트 속성들에 대해서, 오브젝트의 리터럴 값은 persistent 오브젝트의 상태와 함께 저장된다.

## 4) OID 와 ID 값

Caché 에서는 “범용” 오브젝트 식별자(OID)와 “로컬” 오브젝트 식별자(ID)를 구별하는데 있어 추가 정의를 보면, OID 값은 데이터베이스에 저장된 오브젝트를 유일하게 식별하는 값으로서, 특히 ID 값과 함께 클래스명으로 구성되어 있으며, ID 값은 특정 범위(extent)에 속하는 오브젝트의 유일한 식별자이다. 여기서 범위(extent)는 특정 클래스의 모든 인스턴스 또는 클래스와 하위클래스의 모든 인스턴스들을 포함하는 관련 오브젝트의 집합을 말한다.

Caché 는 persistent 오브젝트를 참조하는데 두 가지 메소드를 제공하는데, 하나는 OID 로, 다른 하나는 ID 값을 사용한다. 다음은 OID 값을 사용하는 메소드이며,

```
SET object = ##class(MyApp.Person).%Open(oid)
```

아래는 ID 값을 사용하는 메소드이다:

```
SET object = ##class(MyApp.Person).%OpenId(id)
```

보는 바와같이 ID 값을 이용하는 메소드는 메소드명에 **Id** 를 추가하였다. 실제로 대부분의 응용프로그램에서는 ID 값과 함께 메소드 **%OpenId()** 를 사용하며, 이는 해당 오브젝트가 어떤 범위(오브젝트 집합)에 속하는지 모르고 오브젝트를 찾아보는 경우는 거의 없기 때문이다.

## 5) OREF 와 참조 카운트

메모리에서 참조되는 오브젝트 참조(OREF)는 자동적으로 참조 조회수를 관리하는데 참조 조회수란 하나의 오브젝트에 대하여 참조하고 있는 아이템(변수 또는 오브젝트) 개수를 말한다. 오브젝트를 참조하기 위해 해당 오브젝트를 변수 또는 오브젝트 프로퍼티에 OREF 값을 설정하게 되면, 참조 조회수가 자동으로 하나씩 증가하며, 참조가 종료되면(즉, 설정값을 갖고 있던 변수가 삭제되거나 동일 변수에 새로운 값이 설정되는 경우 참조 조회수는 하나씩 줄어든다. 마지막으로 조회수가 0 이 되면, 오브젝트는 자동적으로 메모리로부터 제거되며, **%OnClose** 메소드가 실행된다.

다음의 예를 보면,

```
Method Test()
{
    SET person1 = ##class(Sample.Person).%OpenId(1)

    SET person2 = ##class(Sample.Person).%OpenId(2)
    ...
    KILL person1
}
```

Sample.Person 인스턴스를 생성하여 해당 참조값을 변수 person1 에 설정한다. 그 다음에 Sample.Person 의 다른 인스턴스를 생성하여, person2 에 그 값을 설정하게 되면, 이 시점에서 Sample.Person 참조 조회수는 2 가 된다. 그 후, 변수 person1 에 대하여 KILL 명령을 실행하면, person1 이 갖고 있던 Sample.Person 의 참조가 없어지고, 참조 조회수는 하나 감소(즉, 1)된다. 메소드 Test() 가 실행을 마치고 종료되면 person2 가 갖고 있던 참조값도 제거되어 조회수는 0 이 되며, 이 때, **%OnClose()** 메소드가 자동 실행되어 메모리에서 Sample.Person 오브젝트 인스턴스를 제거한다.

ObjectScript 의 **\$IsObject** 함수를 사용하여, 주어진 변수가 유효한 OREF 값을 가지고 있는지 확인할 수 있다:

```
SET obj = ##class(Sample.Person).%OpenId(1)
Write "Is (1) an object? ",$IsObject(obj),!
```

## 2. 클래스 유형

Caché 는 기능별로 다양한 클래스 유형을 지원하고 있는데, 크게 데이터 타입 클래스와 오브젝트 클래스로 나누어진다. 데이터 타입 클래스는 문자열, 정수, 날짜등과 같은 문자값을 표현하며, 다른 오브젝트의 프로퍼티를 생성하는데 사용된다. 데이터 타입 클래스는 자체 프로퍼티를 가지고 있지 않을 뿐만 아니라 인스턴스를 생성할 수도 없다.

오브젝트 클래스는 자신의 프로퍼티를 가질 수 있어 인스턴스도 생성할 수 있다. 대부분의 오브젝트 클래스는 시스템 클래스 **%RegisteredObject** 의 하위 클래스로서,

- 인스턴스 생성시, 오브젝트 클래스의 프로퍼티를 위한 시스템 메모리 자동 할당.
- 인스턴스 생성시, 오브젝트 클래스에 대한 오브젝트 참조(OREF) 자동 생성.
- 다형성(polymorphism) 지원

오브젝트 클래스는 클래스의 데이터베이스 기능에 따라, **비상주(transient)** 오브젝트, **지속형(persistent)** 오브젝트, **시리얼(serial)** 오브젝트로 나누어진다. 비상주(transient) 오브젝트(**%RegisteredObject** 상속)는 저장되지 않고 메모리에서만 존재하는 클래스이며, 지속형(persistent) 클래스(**%Persistent** 상속)는 독자적으로 데이터베이스에 저장된다. 한편 시리얼(serial) 오브젝트(**%SerialObject** 상속)는 다른 오브젝트에 임베디드되는 클래스이다 – 즉, 다른 persistent 오브젝트에 내장되어야만 데이터베이스에 저장될 수 있다.

### 1) 비상주 (transient) 오브젝트 클래스

**%RegisteredObject** 클래스로부터 상속된 클래스들의 인스턴스는 레지스터 또는 비상주(transient) 오브젝트라고 한다. 이러한 오브젝트는 메모리에서의 기능을 제어하기 위한 빌트-인 메소드들을 갖추고 있다. (대부분의 응용프로그램에서는 **%RegisteredObject** 에서 직접 상속하는 대신, persistent 클래스나 임베디드 클래스를 사용한다).

레지스터 오브젝트의 인스턴스가 생성되면, OREF 를 사용하여 참조된다. 이 시스템에서 주어진 값은 비상주이므로 오브젝트가 메모리에 있을 때에만 존재하며, 서로 다른 호출시에 동일한 값이 설정된다는 보장은 없다.

레지스터 오브젝트는 자신의 값을 저장하기 위해 시스템이 할당한 메모리를 사용하며, 다형성 속성을 지원한다.

## 2) 지속 (persistent) 오브젝트 클래스

%Persistent 클래스에서 상속된 레지스터 오브젝트를 persistent 오브젝트라고 하며 데이터베이스에 저장되는 오브젝트이다.

Persistent 오브젝트를 데이터베이스에서 메모리로 로딩할 때, 해당 오브젝트에서 참조한 다른 오브젝트들은 같이 로딩되지 않는다. 참조된 오브젝트들은 해당 오브젝트가 조회되는 시점에 비로소 메모리로 로드되는데 이를 스위즐링(swizzling) 또는 지연 로딩(lazy loading)이라고 한다.

Persistent 오브젝트가 하나의 프로퍼티로 사용되면, 이를 참조 프로퍼티라고 한다. 예를 들어, Doctor 와 Patient 가 각각 persistent 클래스라고 한다면, 아래와 같이 Doctor 클래스를 Patient 클래스의 한 프로퍼티로 속성을 정의할 수 있다:

```
Property TheDoc As Doctor;
```

여기서, *TheDoc* 프로퍼티는 프로퍼티 타입으로 Doctor 오브젝트를 갖으며, 이 때 클래스 Doctor 를 참조 프로퍼티라고 한다. 이 프로퍼티가 다음과 같이 참조되는 시점에,

```
SET doc = patient.TheDoc
```

관련 Doctor 오브젝트가 데이터베이스에서 로딩된다. 로컬 변수 doc 는 로딩된 Doctor 오브젝트에 대한 OREF 를 가진다.

참조 프로퍼티는 계속적으로 다른 클래스들을 연쇄적으로 참조할 수 있다:

```
SET location = patient.TheDoc.Hospital.Address.City
```

이 때, 참조하는 클래스중 하나라도 찾고자 하는 오브젝트 인스턴스가 존재하지 않으면(null), 로컬 변수 location 에는 null 값("")이 설정된다.

## 3) 시리얼(serial) 오브젝트 클래스

%SerialObject 에서 상속된 레지스터 오브젝트를 시리얼(serial) 오브젝트라고 한다. 이 오브젝트는 독자적으로 메모리에 존재는 하지만, 데이터베이스에 저장될 때에는 persistent 오브젝트에 내장되어서만 존재하게 된다.

%SerialObject 클래스는 다음과 같은 특징을 갖는 오브젝트를 제공한다:

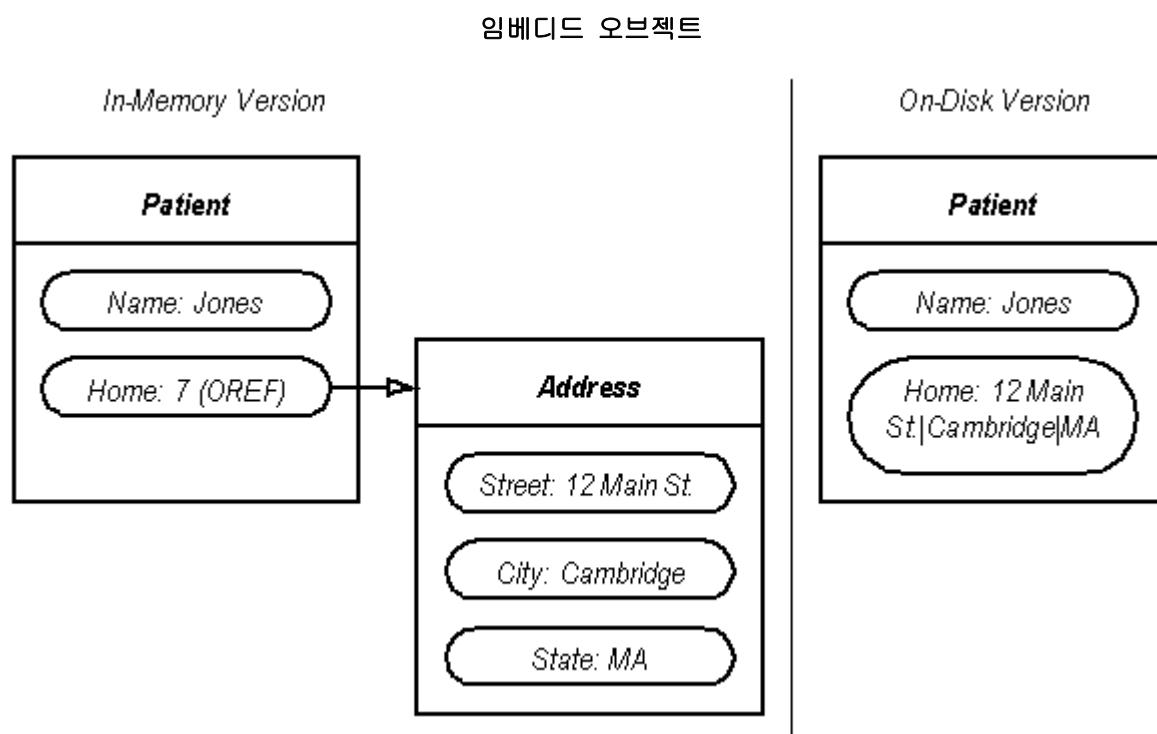
- 오브젝트의 상태를 나타내는 문자열(즉, 현재의 프로퍼티 값)을 생성한다. 이런 문자열의 생성을 오브젝트 순차화(serializing) 라고 한다.
- 시리얼 문자열을 사용한 지연(swizzle) 기능. Persistent 오브젝트와 마찬가지로, 시리얼 오브젝트는 프로퍼티 타입으로 사용될 수 있으나, 디스크상에서의 상황은 persistent

오브젝트와는 다르다. 프로퍼티로 사용되는 경우, 내장 가능 클래스는 임베디드 오브젝트라고 한다.

임베디드 오브젝트는 메모리와 디스크상에서 달리 표현된다:

- 메모리에서의 임베디드 오브젝트는 개별 오브젝트로 존재하며 다른 오브젝트에 대한 참조와 구분이 되지 않는다. 임베디드 오브젝트 속성의 메모리에서의 값은 오브젝트의 메모리에서의 참조값인 OREF이다.
- 반면에, 디스크상에서는, 임베디드 오브젝트 프로퍼티는 해당 오브젝트를 포함하고 있는 오브젝트의 일부분으로 저장된다. 시리얼 오브젝트는 독립된 식별자(OID)를 가지고 있지 않으며, 다른 오브젝트에 의해 참조될 수도 없다. 임베디드 오브젝트 프로퍼티의 값은 시리즈화되어 다른 오브젝트 속성들과 함께 저장된다.

다음 그림은 메모리와 디스크에서의 임베디드 오브젝트 위치를 설명하고 있다.



참고: Caché의 persistent 클래스에서 시리얼 오브젝트의 다형성을 지원하지 않는다.

#### 4) 메모리 와 디스크에서의 임베디드 오브젝트

임베디드 오브젝트를 포함하는 오브젝트의 새 인스턴스가 생성되면, %New 메소드는 자동적으로 메모리에서의 오브젝트 인스턴스를 생성한다.

다음과 같은 Address 클래스를 예로 들어보자:

```
Class MyApp.Address Extends %SerialObject
{
  Property Street As %String(MAXLEN=80);
  Property City As %String(MINLEN=3);
  Property State As %String(MINLEN=2);
}
```

위의 클래스가 정의되면 Address 클래스는 Patient 처럼 persistent 오브젝트 클래스의 임베디드 오브젝트 속성으로 사용된다:

```
Class MyApp.Patient Extends %Persistent
{
  Property Name As %String;
  Property Home As Address;
}
```

그리고 메모리에서, Home 프로퍼티는 다른 오브젝트 프로퍼티처럼 사용할 수 있다.

```
// Open a persistent patient object
Set patient = ##class(MyApp.Patient).%OpenId(22)

// Get the patient's city and state from its Home address
Set city = patient.Home.City
Set state = patient.Home.State
```

## 5) 데이터 타입 클래스

데이터 타입 클래스는 리터럴(literal) 값을 정의하고 제어한다. 오브젝트 클래스와는 달리, 데이터 형식 클래스는 독립된 식별자를 갖지 않으며 인스턴스를 생성할 수도 없다. 또한 데이터 타입 클래스는 오브젝트의 프로퍼티로만 존재하며, 자체적으로 프로퍼티를 갖지 못한다.

데이터 타입 클래스는 *ClassType* 키워드값으로 “**datatype**” 을 갖으며, 데이터 타입 인터페이스가 갖추어야하는 특정의 메소드들을 구현한다. 이러한 인터페이스에는 정당성 검사와 SQL 실행을 위한 오퍼레이션들도 포함되어 있다.

예를 들어, 다음의 Patient 클래스는 시스템 데이터 타입 **%String** 을 사용하는 프로퍼티 *Name* 을 정의하고 있다:

```
Class MyApp.Patient Extends %Persistent
{
  Property Name As %String;
}
```

`Name` 프로퍼티는 `Patient` 오브젝트의 특정 인스턴스내에 값으로서 존재하며, 오브젝트의 속성으로만 참조 가능하다:

```
SET name = pat.Name
```

### 3. 상속

Caché 오브젝트 모델은 “*Super*” 키워드를 사용하여 현재 정의되어 있는 클래스들을 확장(상속)시킬 수가 있다. 이 때, 이들 클래스들을 상속받아 정의되는 클래스를 **하위클래스**라고 하며, 상속된 클래스를 **상위클래스**라고 한다.

Caché 오브젝트 모델에서, 클래스 정의는 *Super* 키워드를 사용하여 상속받는 상위클래스들의 목록을 보유하며, Caché Studio에서의 클래스 코드에서는 “*Extends*” 키워드를 사용하여 동일한 상위클래스들의 목록을 정의한다:

```
Class User.MyClass Extends %Persistent
{
}
```

클래스는 프로퍼티, 메소드, 클래스 파라미터, 클래스 키워드등을 포함하여 참조한 상위클래스의 모든 속성들을 상속한다. *Final*로 설정된 속성들을 제외하고, 하위클래스는 상속된 컴포넌트들의 대부분의 속성들을 재정의(삭제는 허용안됨)할 수 있다.

또한, 상속받은 클래스의 메소드와 프로퍼티와는 별도로, 새로운 메소드와 프로퍼티들을 하위클래스에 추가할 수도 있다.

예를 들어, 다음과 같이, 상위클래스 `Person`에 대하여,

```
Class MyApp.Person Extends %Persistent
{
    Property Name As %String;
    Property DOB As %Date;
}
```

이를 상속받은 하위클래스 `Employee`를 다음과 같이 정의할 수 있다:

```
Class MyApp.Employee Extends Person
{
    Property Salary As %Integer;
    Property Department As %String;
}
```

여기에서, Employee 클래스는 자체적인 프로퍼티, Salary 와 Department 를 갖고 있으면서도, Person 클래스로부터 상속받은 프로퍼티 Name 과 DOB 를 자신의 프로퍼티처럼 사용할 수 있다:

```
SET x = ##class(MyApp.Person).%OpenId(id)
WRITE x.Name
WRITE x.Salary
```

## 1) 다중 상속

Caché 오브젝트 모델은 다중 상속 기능을 제공하기 때문에, 클래스는 복수의 상위클래스들로부터 기능과 클래스 타입을 상속받을 수 있다. 다중 상속을 정의하려면, 클래스의 **Super** 키워드에 상속받으려는 클래스들의 목록을 콤마로 구별하여 설정한다.

클래스 컴파일시, 클래스 컴파일러는 정의된 상위클래스의 목록을 참조하여 클래스의 구성 요소(클래스 키워드를 제외한 클래스 파라미터, 메소드, 프로퍼티등)들을 설정한다. 정의된 상위클래스들 가운데 동일한 이름의 구성요소가 존재하는 경우, 목록중 나중에 정의된 상위클래스가 우선한다.

그렇지만, 클래스 키워드 값은 대체되지 않기 때문에, 하위클래스는 첫번째 정의된 상위클래스의 키워드 값만을 상속한다.

예를 들어, 다음과 같이 클래스 X 가 클래스 A, B, C 를 상속한다고 하면,

```
Class X Extends (A, B, C) {  
}
```

클래스 X 의 파라미터, 프로퍼티, 메소드는 우선 클래스 A (첫 상위 클래스 목록), 그 다음 B, 그리고 최종적으로 C 로부터 상속된다. 만약 클래스 B 에 A 클래스로부터 상속된 구성 요소와 동일한 이름의 멤버가 존재하는 경우, B 클래스의 멤버가 A 로부터 상속받은 구성요소를 대체한다. 동일한 방식으로 C 클래스가 A, B 에 우선한다. 그렇지만 클래스 X 의 키워드는 A 클래스가 우선한다.

## 4. 클래스 컴파일

Caché 클래스 정의는 Caché 클래스 컴파일러에 의해 어플리케이션 루틴으로 컴파일된다. 컴파일되기 전에는 클래스를 사용할 수 없다.

Caché 클래스 컴파일러는 C++, Java 같은 다른 프로그램 언어에 사용하는 컴파일러와는 다음과 같은 차이점이 있다: 첫째, 컴파일 결과는 파일 시스템이 아닌 공유 저장소(데이터베이스)에 저장된다. 둘째, Caché 컴파일러는 persistent 클래스에 대한 지원을 자동적으로 제공한다.

클래스 컴파일러는 다음의 절차에 따라 수행된다:

1. 의존성(dependency) 목록 작성 – 먼저 컴파일되어야 하는 클래스들을 결정. 컴파일 옵션에 따라 마지막을 컴파일한 후 수정된 의존성 목록(dependency)이 컴파일 된다.
2. 상속 결정 – 어떤 구성요소들이 상위 클래스로부터 상속되는지를 결정한다. 이러한 상속 정보는 추후 사용을 위해 클래스 사전(dictionary)에 저장한다.
3. persistent, serial 클래스인 경우, 데이터베이스에 저장할 오브젝트의 스토리지 구조를 결정하고 클래스의 SQL 인터페이스를 위해 필요한 런타임 정보를 생성한다.
4. 클래스에 의해 정의되었거나 상속된 메소드 생성기를 실행한다.
5. 클래스의 실행 코드를 갖는 하나 이상의 Caché ObjectScript 루틴을 생성한다. 클래스 컴파일러는 정의된 언어(Caché ObjectScript 또는 Basic)에 기반하여 메소드들을 분류하고 해당 언어로 정의된 메소드들에 대한 루틴을 생성한다.

클래스 컴파일러의 옵션 “생성된 소스 코드 유지” 을 지정하면, Studio에서 “보기 -> 기타 코드 보기” 메뉴를 통해 생성된 루틴의 소스를 볼 수 있다.

6. 생성된 모든 루틴을 실행 가능한 코드로 컴파일 한다.
7. 클래스에 대한 설명(class descriptor)을 생성한다. 클래스 설명은 루틴으로 저장되는 특별한 데이터 구조로서 클래스 및 해당 구성요소들을 지원하는데 필요한 모든 런타임 정보를 포함한다.

클래스 컴파일러를 실행하는 방법은

- Caché Studio의 빌드 메뉴에 있는 옵션 사용
- Caché 터미널에서 **%SYSTEM.OBJ** 클래스의 **Compile** 메소드를 사용

```
DO $System.OBJ.Compile("MyApp.MyClass")
```

테이블을 생성하기 위해 SQL DDL 명령문을 사용하는 경우, 해당 테이블에 대응하는 persistent 클래스를 컴파일하기 위해 클래스 컴파일러를 자동 호출한다.

또한 **Compile** 메소드는 추가적인 컴파일 옵션들을 메소드 인수로서 제공받을 수 있다. 옵션 목록은 다음과 같이 확인할 수 있다:

```
DO $System.OBJ.ShowFlags()
```

# 제 5 장 Caché SQL

## 1. Caché SQL 소개

- 1) 아키텍처
- 2) 기능
- 3) 상호 운용성

## 2. Caché SQL 기초

- 1) 테이블
- 2) 쿼리
- 3) 권한
- 4) 조합(Collation)
- 5) SQL 쉘
- 6) SQL과 시스템 관리 포탈

## 3. 저장 프로시저의 정의와 사용법

- 1) 개요
- 2) 저장 프로시저 정의
- 3) 저장 프로시저 사용 방법

## 4. Caché SQL 참조

- 1) 집계(Aggregate) 함수
- 2) 환경 설정
- 3) 데이터 형식
- 4) 날짜와 시간 구조 (Date and Time Constructs)
- 5) 디풀트 사용자명과 패스워드
- 6) 필드 제약
- 7) 예약어
- 8) 문자열 제어

## 1. Caché SQL 소개

Caché SQL은 Caché 데이터베이스에 저장된 데이터에 대한 완벽한 표준 관계형 액세스를 제공한다.

Caché SQL은 다음과 같은 특징들을 구현한다:

- **고성능 및 확장성:** Caché SQL은 다른 관계형 데이터베이스 제품보다 우수한 성능과 확장성을 구현할 뿐만 아니라 일반 노트북에서부터 최신 하이엔드 멀티프로세싱 시스템에 이르는 다양한 하드웨어와 운영 체계에서 사용할 수 있다.
- **Caché 오브젝트 기술과의 통합:** Caché 오브젝트 기술과 완벽하게 통합되어 있으며, 따라서 데이터 접근방식의 차이로 인한 성능 저하없이 관계형과 오브젝트 인터페이스를 동시에 혼합하여 사용할 수 있다.
- **효율적인 유지 관리:** 다른 관계형 데이터베이스와는 달리, Caché 응용프로그램에서는 서비스되고 있는 어플리케이션에서의 주기적인 인덱스 재생성 및 테이블 압축 작업이 필요하지 않다.
- **표준 SQL 쿼리 지원:** SQL-92 표준 문법과 명령어를 지원한다. 대부분의 경우, 기존의 관계형 어플리케이션을 쉽게 Caché로 마이그레이션하여, 곧바로 Caché의 고성능 오브젝트 기술의 장점을 이용할 수 있다.

Caché SQL은 다음과 같은 다양한 목적에 사용된다:

- **오브젝트 및 웹 기반 어플리케이션:** Caché 오브젝트와 Caché CSP (Caché Server Page) 어플리케이션에서의 조회와 탐색등과 같은 데이터베이스 작업을 효율적으로 수행하기 위해 SQL 쿼리를 사용할 수 있다.
- **온라인 트랜잭션 처리:** 신규 등록 및 갱신과 같은 작업뿐만 아니라 트랜잭션 처리 응용프로그램에서 필요로하는 특화된 쿼리에 대해 뛰어난 성능을 발휘한다.
- **비즈니스 인텔리전스 및 데이터 웨어하우스:** Caché의 다차원 데이터베이스 엔진과 비트맵 인덱스 기술의 결합으로 데이터 웨어하우스 형태의 프로그램에 효율적이고 뛰어난 성능의 어플리케이션을 구현할 수 있다.
- **동적 쿼리 및 리포트:** Caché SQL에 포함되어 ODBC 및 JDBC 드라이버를 사용하여, 범용 리포트와 쿼리 도구에 연결 가능하다.
- **엔터프라이즈 어플리케이션 통합:** Caché SQL Gateway를 이용하여 ODBC 또는 JDBC를 지원하는 외부 관계형 데이터베이스에 저장된 데이터에 접근할 수 있으며, Caché 어플리케이션의 다양한 소스의 데이터와 쉽게 통합할 수 있다.

## 1) 아키텍처

Caché SQL의 핵심은 다음의 요소로 구성되어 있다:

- **통합 데이터 사전:** 클래스 정의의 형태로 저장된 메타 정보의 저장소로서, Caché는 통합 사전에 저장된 모든 persistent 클래스에 대응하는 관계형 테이블 정의를 자동 생성한다.
- **SQL 프로세서 및 옵티마이저(optimizer, 최적화 처리 장치):** SQL 쿼리를 분석하는 프로그램 그룹으로서, 주어진 쿼리에 대한 (비용 기반) 최적의 검색 플랜을 정의하고, 쿼리를 실행할 코드를 생성한다.
- **Caché SQL 서버:** Caché ODBC와 JDBC 드라이버를 통한 모든 통신을 제어하는 Caché 서버 프로세스로서 자주 사용되는 쿼리의 캐시도 관리한다; 동일한 쿼리가 계속적으로 요청되고 반복 실행 되는 경우, 옵티마이저에 의해 동일한 쿼리를 매번 분석하는 대신 캐시된 해당 쿼리를 바로 실행할 수 있다.

## 2) 기능

Caché SQL은 다음과 같이 표준 관계형 기능들을 모두 지원하고 있다:

- 테이블과 뷰 정의 – DDL(Data Definition Language).
- 테이블과 뷰에 대한 쿼리 실행 – DML(Data Manipulation Language).
- INSERT, UPDATE, DELETE 를 포함한 트랜잭션 실행. 동시에 작업이 실행되면, Caché SQL은 행 단위의 잠금(lock)을 사용한다.
- 보다 효율적인 쿼리를 위한 인덱스 정의.
- 사용자 정의 타입을 비롯한 다양한 데이터 타입 사용.
- 사용자 및 Role 정의 및 권한 부여.
- 외부 키 및 기타 무결성 제약조건 정의.
- INSERT, UPDATE, DELETE 트리거 정의
- 저장 프로시저 정의 및 실행.
- 여러 형태로의 데이터 출력 형식 변환 – 클라이언트 액세스를 위한 ODBC 모드와 (CSP 페이지 등과 같은) 서버 기반 어플리케이션에서 사용되는 Display 모드가 있다.

### SQL-92 호환성

Caché SQL에서는 다음의 경우를 제외하고, 엔트리 레벨 표준 SQL-92 를 완벽하게 지원한다:

- 테이블 정의에 추가적인 CHECK 제약 조건 추가는 지원하지 않는다.
- SERIALIZABLE 격리 수준(isolation level)은 지원하지 않는다.
- SQL 표현식에서의 산술 연산자 순위는 표준 SQL-92 와 다르다. Caché SQL은 산술 표현식

계산에서 일반적인 연산자 우선 순위를 따르지 않고, 왼쪽에서 오른쪽으로 순차적으로 처리한다. 이는 Caché ObjectScript에서 사용하는 규약과 동일한 방식이다. 예를 들어,  $3+3*5=30$ 이 되며, 우선 순위가 필요한 경우, 괄호를 사용한다;  $3+(3*5)=18$ .

- 식별자들은 대소문자를 구분하지 않는다. 반면에, 표준의 경우, 대소문자를 구분한다.
- HAVING 절에 포함된 서브 쿼리에서, HAVING 절에서 사용 가능한 집계(aggregates) 함수들을 참조할 수 있는 기능은 지원되지 않는다.

## 확장

Caché SQL은 다수의 유용한 확장 기능들을 지원하는데, 대부분의 기능들이 데이터 액세스를 위해 오브젝트와 관계형 인터페이스를 동시에 사용할 수 있도록 구현하는 것과 관계된다. 주요 확장 기능은 다음과 같다:

- 사용자 정의 데이터 타입 및 함수 지원
- 오브젝트 참조를 위한 인터페이스
- 하위클래스 파생 및 상속
- 외부 데이터베이스에 저장된 테이블에 대한 쿼리
- 최고의 성능을 위한 테이블 저장 구조 제어 메커니즘

## 3) 상호 운용성

Caché SQL은 외부 응용프로그램 및 소프트웨어 도구들과의 SQL 인터페이스를 통해 상호 액세스를 가능케하는 다양한 방법들을 지원하고 있다.

### JDBC

Caché는 표준 호환 레벨 4 (순수 자바 코드) JDBC 클라이언트를 포함하고 있다. Caché JDBC 드라이버에는 다음과 같은 특징들을 구현한다:

- 고 성능
- 순수 자바 구현
- 유니코드 지원
- 스레드 안전성

Caché JDBC는 JDBC 환경의 모든 도구, 응용프로그램, 및 개발 환경 등을 지원한다.

## ODBC

Caché SQL의 C 언어 호출 레벨 인터페이스(CLI)는 ODBC이다. 다른 데이터베이스 제품과는 달리, Caché ODBC 드라이버는 어떠한 독점적 인터페이스에 기반하지 않은 네이티브 드라이버이다. Caché ODBC 드라이버는 다음과 같은 특징들을 구현한다:

- 고 성능
- 이식성
- 네이티브 유니코드 지원
- 스레드 안전성

Caché ODBC는 ODBC 환경의 모든 도구, 응용프로그램, 및 개발 환경 등을 지원한다.

## 임베디드 SQL

Caché SQL은 임베디드 SQL을 지원한다. 이는 Caché Objectscript 루틴이나 메소드에서 SQL 문을 직접 정의하고 실행하는 기능을 의미한다.

임베디드 SQL은 Caché의 오브젝트 액세스 기술과 접목하여 사용함으로써 어플리케이션 개발 효과 및 로직 구현을 극대화할 수 있다. 예를 들어, 다음의 메소드는 SQL 쿼리를 사용하여 Product 테이블에서 SKU 값을 갖는 오브젝트 ID를 검색하고, 오브젝트 인터페이스를 사용하여 검색된 ID의 오브젝트 인스턴스를 생성한다:

```
ClassMethod FindBySKU(sku As %String)
{
    &sql(SELECT %ID INTO :id FROM Product WHERE SKU = :sku)

    If (SQLCODE = 0) {
        // product에 상세정보 표시를 요청한다
        Set product = ##class(Product).%OpenId(id)
        Do product.DisplayDetails()
    }
}
```

## 동적 SQL

Caché 는 표준 라이브러리의 멤버로서 **%Library.ResultSet** 클래스를 제공하는데, 이 클래스를 이용하여 동적(즉, 실시간에 정의된) SQL 쿼리 실행할 수 있다. 동적 SQL은 Caché ObjectScript 와 Basic 메소드로 사용할 수 있다. 다음은 동적 SQL 쿼리를 사용하여 가격 별로 주문된 제품 목록을 출력하는 메소드이다:

```

ClassMethod ShowByPrice() [ language = basic ]
{
    rs = New %Library.ResultSet()
    rs.Prepare("SELECT SKU, Name FROM MyApp.Product ORDER BY Price")
    rs.Execute()

    While (rs.Next())
        PrintLn rs.Data("SKU") & ":" & rs.Data("Name")
    Wend
}

```

## 2. Caché SQL 기초

여기에서는 표준 SQL에서 다루어지지 않거나, Caché 통합 데이터 아키텍처와 관련된, Caché SQL의 특징에 대한 개요를 설명한다. SQL에 대한 사전 지식을 전제로 한 것으로, SQL 개념이나 기본 문법은 다루지 않는다.

이 장에서 다룰 주요 항목은 다음과 같다:

- 테이블
- 쿼리
- 권한
- 조합
- SQL 셀 인터페이스
- SQL과 시스템 관리 포탈

### 1) 테이블

Caché SQL의 데이터는 테이블 단위로 표현된다. 각 테이블은 여러 개의 컬럼으로 구성되며 데이터를 담고 있는 레코드(행)들을 갖게된다.

테이블에는 두 가지 형이 있는데, 하나는 데이터를 실제로 가지고 있는 **기본(base)** 테이블로서 일반적으로 그냥 테이블이라 불리는 것과 하나 이상의 테이블을 참조하여 데이터를 나타내는 논리적 테이블인 **뷰(View)**이다.

테이블에 대한 검색을 보다 효율적으로 하기 위하여 **인덱스**를 정의하며, 테이블간의 데이터 참조를 위하여 해당 테이블들에 대한 **외부 키**와 **트리거**를 정의할 수도 있다.

## 스키마

SQL 스키마는 관계형 테이블의 집합을 구성하는 방법을 제공한다. 하나의 응용프로그램에서 여러 개의 스키마를 구성할 수 있는데, 이는 동일 이름을 갖는 테이블간의 충돌을 방지할 수 있다. SQL 스키마는 Caché 오브젝트 모델의 패키지와 유사하다. 특정 네임스페이스내의 모든 스키마는 보기 위해서는,

1. 시스템 관리 포탈에서 메뉴 SQL 을 선택한 후, “SQL 스키마 검색” 를 클릭하면 현재 선택된 네임스페이스에 존재하는 모든 스키마의 목록이 나타난다.
2. 다른 네임스페이스의 스키마를 보려면, 페이지 왼쪽의 목록에서 원하는 네임스페이스를 선택한다.
3. 이 페이지에서 스키마의 컨텐츠를 확인할 수 있는데 페이지에 출력된 테이블의 첫번째 컬럼은 현재 네임스페이스에 정의된 스키마명이다. 각 스키마에 정의된 Caché 쿼리, 저장 프로시저, 테이블, 뷰를 포함하고 있다면, 이들에 대한 링크가 테이블의 각 컬럼에 표시된다. 컬럼중 하나의 링크를 클릭하면, 해당 아이템의 목록이 표시된다. 참고로 스키마명을 클릭하면 테이블 목록이 표시된다.

## 2) 쿼리

Caché SQL에서 테이블의 데이터는 SQL 쿼리문을 사용하여 변경하거나 확인할 수 있다. 일반적으로, 쿼리는 두 가지 형태로 분류되는데, 하나는 데이터를 검색하는 쿼리(SELECT)이고 다른 하나는 데이터를 변경하는 쿼리(INSERT, UPDATE, DELETE)이다.

다음은 SQL 쿼리문을 정의하고 실행할 수 있는 방법들이다:

- Caché ObjectScript 및 오브젝트에서의 임베디드 SQL
- Caché ObjectScript 및 오브젝트에서의 동적 SQL
- 다양한 외부 환경에서의 ODBC 또는 JDBC 인터페이스

## 3) 권한

Caché SQL 은 테이블, 뷰 등과 같은 SQL 아이템들로의 액세스 범위를 제한할 수 있는 권한제어를 제공하여, 해당 사용자 및 역할(Role)만이 테이블에 대한 특정 작업(읽기, 쓰기등)을 할 수 있도록 정의한다.

## 4) 조합(Collation)

조합은 값의 SQL 환경에서의 데이터의 정렬 순서를 설정하는 것으로서 CREATE TABLE 및 ALTER TABLE 에서의 컬럼 정의에 포함시키거나, 또는 데이터를 특정 조합 타입으로 변환하는 조합 함수를 사용하여 정의한다. 예를 들어, SQLUPPER 조합은 문자열을 대문자로 변환시킨다.

### 주요 조합 타입

조합 타입	내 용
EXACT	변형 없이 있는 그대로의 상태(문자뒤 여백 및 대소문자 구별 유지)를 비교하여 정렬한다. 이 타입은 숫자로 시작하는 문자열(예를 들어, '123', '123abc', '-123abc' 등등)에는 적용하지 말아야 한다.
MVR	변형 없이 있는 그대로의 상태(문자뒤 여백 및 대소문자 구별 유지)를 비교하여 정렬한다. EXACT와 MVR은 NULL 처리를 제외하고는 동일하다. MVR은 MultiValue 데이터베이스 시스템과의 호환성을 위해 제공된 것이다.
SQLSTRING	대소문자 구분하여 문자열 정렬. 데이터를 문자열로 변환하고 문자열 끝에 따라붙은 공백(여백, 탭 등)들을 없애며, 문자열의 시작에 한 칸의 여백을 추가한다. 여백(스페이스, 탭등)만을 갖는 데이터는 SQL의 빈 문자열로 정렬시킨다.
SQLSTRING(n)	SQLSTRING 과 동일하나 처음 n 개의 문자만을 비교하여 정렬한다. 이 타입은 인덱스 작업 및 긴 문자열 정렬을 향상시키는데 사용한다.
SQLUPPER	대소문자 구별없이 문자갑 정렬(모든 문자는 대문자로 변환). 모든 값의 끝에 붙은 여백(스페이스, 탭 등)을 없애고 문자열 앞에 하나의 공백 스페이스를 첨가한 문자열로 변환한다. 문자열 앞에 첨가된 공백 문자는 NULL 또는 수치값의 문자열 변환을 강제한다. SQLUPPER는 여백(스페이스, 탭등)만을 갖는 값은 SQL 빈 문자열로 정렬시킨다. SQLUPPER는 ALPHAUP, STRING, UPPER 타입 보다 선호되어 사용된다.
SQLUPPER(n)	SQLUPPER 와 동일하지만 처음 n 개의 문자만을 비교한다(n 은 양의 정수). 긴 문자열의 인덱싱 및 정렬 작업 개선에 주로 사용한다.
SPACE	SQLSTRING 과 동일하지만 문자열 끝에 포함된 공백을 없애지 않는다.

선택적으로, 조합 타입(Collation) 파라미터 키워드의 맨 앞에 퍼센트(%) 문자를 사용하여, 다음과 같은 SQL 연산에(작업) 필드의 조합 값으로 사용한다:

- 필드가 ORDER BY 절에서 사용될 때
- SQL 비교 연산자(=, >, <, o, BETWEEN 등)를 필드와 함께 사용할 때
- %STARTSWITH 연산을 필드에 대하여 사용할 때
- MIN, MAX, DISTINCT ,GROUP BY 연산을 필드와 함께 사용할 때 – UNION은 DISTINCT 연산을 포함한다.

필드의 조합은 다음과 같이 정의된다:

1. 필드의 데이터 타입은 기본 조합을 갖는다. %String과 같은 문자열 데이터 타입의 기본값은 SQLUPPER 이다.
2. COLLATION 파라미터에 값을 설정하여 필드의 조합을 변경할 수 있다. 예를 들어, 다음 클래스는 EXACT 조합을 사용하는 Name 프로퍼티를 정의하고 있다:

```

Class MyApp.Patient Extends %Persistent [ClassType = persistent]
{
  /// Patient name
  Property Name As %String(COLLATION = "Exact");

  /// ...
}

```

**참조:** 일반적으로 대부분의 경우, 시스템이 제공하는 디폴트 정렬 타입이 어플리케이션에 적절히 적용되기 때문에 사용자는 조합 타입의 설정에 대해서 그리 신경쓰지 않아도 된다.

#### 추가 참고 사항:

- 만약 이미 데이터가 존재하는 클래스의 프로퍼티에 대한 조합을 변경하는 경우, 해당 프로퍼티의 인덱스는 반드시 재생성하여야 한다.
- LIKE 비교 연산자에 조합을 사용하지 않는다.
- (LEFT, SUBST 과 같은 스칼라 스트링 함수를 사용하는) 문자열 표현식은 EXACT 조합으로 그 결과를 산출한다.

SQL 조합에 대한 추가적인 정보는 “[SQL 조합과 NLS 조합](#)” 절을 참고하기 바란다.

#### 인덱스 값의 조합

디폴트로, 주어진 프로퍼티의 인덱스는 프로퍼티 데이터의 조합 타입을 사용한다. 예를 들어, %String 타입의 프로퍼티 *Name* 을 다음과 같이 정의한 경우,

```

Class MyApp.Person Extends %Persistent [ClassType = persistent]
{
  Index NameIDX On Name;

  Property Name As %String;
}

```

*Name*의 데이터 타입 %String 의 조합은 SQLUPPER 이기 때문에 **Person** 테이블에 존재하는 다음과 같은 데이터는,

ID	Name
1	Jones
2	JOHNSON
3	Smith
4	jones
5	SMITH

다음과 같은 인덱스 값을 갖는다:

Name	ID(s)
JOHNSON	2
JONES	1,4
SMITH	3,5

SQL 엔진은 Name 필드에 대한 ORDER BY 나 비교 연산에 이 인덱스를 직접 사용할 수 있다.

만약, 다음과 같이 인덱스 정의에서 **As** 절을 이용하여 해당 인덱스의 조합값을 변경하는 경우,

```
Class MyApp.Person Extends %Persistent [ClassType = persistent]
{
    Index NameIDX On Name As Exact;

    Property Name As %String;
}
```

NameIDX 인덱스는 EXACT(조합되지 않은) 형식으로 저장하게 된다. 예를 들어, 위 Name 의 데이터에 대한 인덱스는 다음과 같다:

Name	ID(s)
JOHNSON	2
JONES	1
jones	4
SMITH	5
Smith	3

일반적으로, 필드의 조합 타입과 다른 값으로 인덱스 조합을 변경하여서는 안되며, 만약 부득이 다른 조합이 필요한 경우, 해당 프로퍼티 레벨에서 정의하여, 인덱스는 프로퍼티와 동일한 조합을 사용하도록 하는 것이 좋다.

복수의 프로퍼티들로 인덱스를 정의하는 경우, 포함된 프로퍼티 각각에 대하여 개별적으로 조합을 설정할 수 있다:

```
Index MyIDX On (Name As Exact, Code As SQLString);
```

## 레거시 조합 함수

Caché에서는 이전 버전과의 호환을 위해 **%ALPHAUP**, **%UPPER**, **UPPER**, **%STRING** 등과 같은 이전 조합 함수를 지원한다.

## SQL 조합과 NLS 조합

또한, Caché는 **\$ORDER** 함수에서의 사용을 위한 조합을 제공하는데, 이 값은 설치된 NLS로 케일에 따른다.

일반적으로, NLS 조합에서는, 각 NLS 로케일, 프로세스, 그리고 특정 글로벌들에 대하여 각각 별개의 조합들을 가질 수 있는 반면에, Caché SQL 에 있어서는 모든 시스템 조합, 프로세스 조합, 그리고 글로벌 조합에 대하여 반드시 동일한 조합을 사용해야 한다.

**참조:** SQL 조합과 NLS 조합을 혼동하지 말아야 한다.

## 5) SQL 쉘

Caché에서는 다음과 같은 다양한 SQL 코드 작성 및 실행 방법을 제공하고 있다:

- Caché ObjectScript 코드에서의 내장(된) SQL 코드 (임베디드 SQL).
- Caché ObjectScript 메소드에서 (와 Caché Basic 메소드로부터 동적 SQL 서술문을 실행하기 위해,) **%ResultSet.SQL** 클래스 또는(나) **%Library.ResultSet** 클래스를 사용하여 동적 SQL 쿼리 실행.
- 시스템 관리 포탈에 SQL 문을 실행할 수 있는 SQL 실행 페이지 제공. (SQL 서술문을 쓰고 실행함)

위의 방법들외에, Caché 터미널로부터 SQL을 실행할 수 있는 SQL 쉘이 추가적으로 제공된다. SQL 쉘을 실행하기 전, 먼저 실행하고자하는 SQL 쿼리의 대상 테이블들이 존재하는 네임스페이스로 이동한다. 그리고 다음의 명령으로 터미널 프롬프트에서 SQL 쉘을 호출한다:

```
DO $SYSTEM.SQL.Shell()
```

SQL 쉘이 호출되면 쉘 프롬프트가(>>) 표시되면 이 프롬프트에 SQL 코드를 입력한 후 엔터키를 누르면 입력된 SQL문의 결과가 출력된다.

SQL 쉘 세션을 종료하고 터미널 프롬프트로 돌아가려면, SQL 프롬프트에서 **EXIT** 명령을 입력한다. 다음은 SQL 쉘 세션의 예이다:

```
USER>ZNSPACE "SAMPLES"
SAMPLES>DO $SYSTEM.SQL.Shell()
SAMPLES>>SELECT Name,Home_State FROM Sample.Person
Djokovic,Josephine W. AK
Klingman,Aviel P. AK
Quine, Sam X. AK
Xiang,Robert C. AL
Roentgen,Alexandria Q. WY
SAMPLES>>SELECT GETDATE()
2009-03-29 11:41:42
SAMPLES>>EXIT
SAMPLES>
```

#### SELECTMODE 설정

Caché SQL 쉘에서, SELECTMODE를 사용하여, 쿼리 데이터를 나타내는 모드를 명시한다.

```
SAMPLES>DO $SYSTEM.SQL.Shell()
SAMPLES>>SET SELECTMODE DISPLAY
selectmode = display (SELECTMODE = DISPLAY)
SAMPLES>>
```

사용할 수 있는 옵션은 DISPLAY, LOGICAL, ODBC, RUNTIME 등으로 LOGICAL이 기본 값이다.

현재의 모드를 정하려면, 값 없이 SELECTMODE를 명시한다.

```
SAMPLES>DO $SYSTEM.SQL.Shell()
SAMPLES>>SET SELECTMODE
selectmode = logical (SELECTMODE = LOGICAL)
SAMPLES>>
```

Caché SQL 쉘에 대한 자세한 설명은 Caché 온라인 문서 [Using Caché SQL](#) 의 Using SQL Shell 부분을 참조하기 바란다.

## 6) SQL과 시스템 관리 포탈

Caché는 시스템 관리 포탈을 통해 데이터를 액세스할 수 있는 SQL 메뉴들을 제공하고 있다. 우선 포탈 메인 페이지의 데이터 관리 컬럼에서 SQL을 클릭하여, SQL 페이지([홈]>[SQL])로

이동한다. SQL 페이지에는 크게 세 종류의 작업으로 분류된다.

### SQL 작업

- SQL 스키마 검색 — SQL 스키마를 검색하고 관리하는 페이지.
- SQL 문 실행 — SQL 쿼리 실행 페이지.
- 새로운 뷰 생성 — 새로운 뷰를 생성하는 페이지.

### SQL 마법사

- 데이터 가져오기 마법사 — 텍스트 파일로부터 Caché 클래스로 데이터를 가져오는 마법사
- 데이터 내보내기 마법사 — Caché 클래스에서 텍스트 파일로 데이터를 내보내는 마법사
- 데이터 이동 마법사 — 외부 소스에서 데이터를 가져오고 이 가져온 데이터를 저장하는데 필요한 Caché 클래스 정의를 생성하는 마법사
- 테이블 링크 마법사 — 외부 소스에 존재하는 테이블이나 뷰를 연결하여 마치 Caché native 데이터처럼 액세스하게 해 주는 마법사
- 프로시저 링크 마법사 — 외부 소스에 정의된 저장 프로시저에 연결하는 마법사
- FileMan 마법사 — Fileman 파일을 InterSystems 클래스로 매핑하는 마법사

### SQL 정보

- SQL 코드 보기 — SQL 오류 코드 목록을 표시
- SQL 예약어 보기 — SQL 예약어 목록을 표시.

## 3. 저장 프로시저의 정의와 사용법

이 장에서는 Caché SQL 에서의 저장 프로시저 정의 및 사용 방법에 대해, 다음 항목을 중심으로 설명한다:

- 저장 프로시저 타입에 대한 개요
- 저장 프로시저 정의
- 저장 프로시저 사용

### 1) 개요

대부분의 관계형 데이터베이스 시스템과 마찬가지로, Caché 에서도 SQL 저장 프로시저를 지원한다. 저장 프로시저(SP)는 데이터베이스에 저장되어 있으면서 (ODBC 또는 JDBC 와 같은)

SQL 인터페이스를 통해 호출 가능한 루틴을 제공하는데, Caché 에서는, ObjectScript 를 이용하여 저장 프로시저를 생성할 수 있다.

다른 관계형 데이터베이스 제품들과의 차이점은, Caché 에서는 클래스 메소드로서 저장 프로시저를 정의한다는 것이다. 사실, Caché 에서의 저장 프로시저는 SQL 인터페이스를 통해 호출할 수 있는 클래스 메소드일 뿐이다. 따라서, 저장 프로시저에서도 프로그램을 위해 제공되는 Caché 오브젝트 기반의 기능들을 모두 활용할 수 있다.

Caché SQL 에는 두 가지 유형의 저장 프로시저가 정의되며, 하나는 메소드 형태로, 다른 하나는 쿼리의 형태로 호출된다. 쿼리 형식의 프로시저는 데이터의 결과 집합이나 여러 행들을 오브젝트 쿼리(쿼리 형식을 지원하도록 설계된 오브젝트)로서 반환하는 것이며, 메소드 유형은 데이터베이스를 간접하고 단일값을 리턴하는 메소드로 구현되는 것이다.

- 결과 집합(result set) 저장 프로시저는 데이터베이스에 대하여 쿼리를 실행하고 레코드의 집합을 반환한다. 이 집합을 결과 집합이라고 한다.
- 메소드 저장 프로시저는 레코드 집합을 반환하지 않는다. 단일값을 반환하는 메소드 저장 프로시저를 저장 함수(stored functions)라고도 한다.

데이터베이스에서 데이터를 반환하는 많은 저장 프로시저가 표준 쿼리 인터페이스를 사용하여 수행될 수 있다. 이러한 접근 방법은 프로시저가 임베디드 SQL 로 작성되는 한 만족스럽게 실행된다.

## 2) 저장 프로시저 정의

저장 프로시저를 정의하는데는 두 가지 방법, 즉 DDL 과 클래스, 가 있다.

### DDL 를 사용하여 저장 프로시저 정의

DDL 의 CREATE METHOD, CREATE FUNCTION, CREATE PROCEDURE 문을 사용하여 메소드 저장 프로시저를 정의할 수 있다.

결과 집합 저장 프로시저는 DDL 의 CREATE QUERY 문을 사용하여 정의한다.

### 클래스를 사용하여 메소드 저장 프로시저 정의

클래스 메소드를 저장 프로시저로 사용할 수도 있는데, 주어진 값을 계산하여 데이터베이스에 저장하는 것과 같이 리턴값을 갖지 않는 경우에 적합하다.

메소드 저장 프로시저를 정의하려면, 다음과 같이 클래스 메소드에 간단히 **SqlProc** 키워드를 정의하면 된다:

```

Class MyApp.Person Extends %Persistent [ClassType = persistent]
{
    /// This procedure finds total sales for a territory
    ClassMethod FindTotal(territory As %String) As %Integer [SqlProc]
    {
        // use embedded sql to find total sales
        &sql(SELECT SUM(SalesAmount) INTO :total
              FROM Sales
              WHERE Territory = :territory
            )

        Quit total
    }
}

```

이 클래스가 컴파일되면, `FindTotal` 메소드가 SQL 에 저장 프로시저 `MyApp.Person_FindTotal` 로 전환된다. 또한 메소드에 `SqlName` 키워드를 설정하여 SQL 에서의 고유한 프로시저 이름을 정의할 수도 있다.

#### 클래스를 사용하여 쿼리 저장 프로시저 정의하기

데이터베이스로부터 데이터를 반환하는 저장 프로시저의 경우, 대부분 표준 쿼리 인터페이스를 통해 수행된다. 이 방식은 프로시저가 임베디드 SQL로 작성되는 만족스럽게 실행된다.

```

Class MyApp.Person Extends %Persistent [ClassType = persistent]
{
    /// This procedure returns a set of persons ordered by name
    Query ListPersons(name As %String = "") As %SQLQuery [ SqlProc ]
    {
        SELECT ID, Name
        FROM Person
        ORDER BY Name
    }
}

```

쿼리를 저장 프로시저로 정의하려면, 해당 쿼리의 `SQLProc` 필드값을 `True` 로 변경하거나, 또는 다음과 같이 쿼리 키워드 “`SqlProc`” 을 쿼리 정의에 추가한다:

```
Query QueryName() As %SQLQuery( ... query definition ... ) [ SqlProc ]
```

이 클래스가 컴파일되면, `ListPersons` 쿼리는 SQL 에 저장 프로시저 `MyApp.Person_ListPersons` 로 전환된다. 쿼리 키워드 `SqlName` 을 사용하여 SQL 에서의 고유한 프로시저 이름을 정의할 수 있다.

SQL로부터 MyApp.Person\_ListPersons 가 호출되면, 쿼리의 SQL 문에서 정의된 레코드 집합이 반환된다.

### 사용자 정의 쿼리

복잡한 쿼리나 일반적인 쿼리 모델에 맞지 않는 저장 프로시저의 경우에는, 쿼리 메소드의 일부 또는 전체를 직접 프로그램하여 쿼리를 정의할 수도 있다.

%SQLQuery 대신에 %Query 타입을 사용하여 쿼리 실행을 보다 용이하게 할 수도 있는데, 타입 %Query 는 개발자가 직접 프로그래밍할 수 있는 4 개의 내부 쿼리 메소드를 제공한다:

- queryname – 쿼리 초기화 및 쿼리 파라미터를 전달한다.
- querynameExecute – 쿼리 실행시 호출, 실행 코드를 작성한다.
- querynameFetch – 생성된 ResultSet 의 각 행을 읽을 때 반복적으로 호출, 각 행의 데이터 값과 형식을 정의한다. AtEnd=1 을 설정하여 마지막 행임을 알린다.
- querynameClose – 필요에 따라 쿼리 종료에 필요한 설정을 수행한다.

이들 메소드들은 qHandle(query handler, %Binary)라는 특수한 서명을 갖고 있는데, qHandle 은 쿼리의 속성과 상태를 보유한 구조체에 대한 포인터로서, Execute 와 Fetch 메소드에서는 참조로, Close 메소드에서는 값으로 전달된다. 다음은 SP1 이라는 사용자 정의 쿼리를 설정했을 때 자동적으로 제공되는 메소드들의 예이다:

```
ClassMethod SP1Close(qHandle As %Binary) As %Status
]
{
    // ...
}

ClassMethod SP1Execute(ByRef qHandle As %Binary,
    p1 As %String) As %Status
{
    // ...
}

ClassMethod SP1Fetch(ByRef qHandle As %Binary,
    ByRef Row As %List, ByRef AtEnd As %Integer=0) As %Status
{
    // ...
}

Query SP1(p1 As %String)
    As %%Query(CONTAINID=0,ROWSPEC="lastname:%String") [sqlproc ]
{
}
```

또한 사용자 정의 데이터들도 qHandle 구조체안에 정의하여 메소드들간에 전달할 수 있다:

```
SET qHandle(1)=oref1,qHandle(2)=data
```

### 3) 저장 프로시저 사용 방법

저장 프로시저 사용에는 다음 두 가지 방법이 있다:

- ODBC, JDBC 또는 임베디드 SQL 에서, SQL **CALL** 명령을 사용하여 저장 프로시저를 호출한다(동적 SQL 에서는 CALL 서술문을 지원하지 않는 데, 이는 Caché 내부에서는 직접 저장 프로시저 메소드 호출을 지원하기 때문이다).
- SQL 쿼리에서 빌트인 함수처럼 저장 함수(즉, 값을 반환하는 메소드 기반의 저장 프로시저)를 사용할 수 있다.

#### 저장 함수(Stored Function)

저장 함수는 단일값을 반환하는 메소드 기반(Result Set 을 리턴하지 않는)의 저장 프로시저를 말한다. 예를 들어서, 다음의 클래스는 주어진 값의 세제곱을 반환하는 저장 함수 Cube 를 정의하고 있다.

```
Class MyApp.Utils Extends %Persistent [ClassType = persistent]
{
    ClassMethod Cube(val As %Integer) As %Integer [SqlProc]
    {
        Quit val * val * val
    }
}
```

저장 함수는 단순히 **SqlProc** 키워드를 설정한 일반적인 클래스 메소드로서 SQL 쿼리문에서 빌트인 SQL 함수인 것처럼 사용할 수 있다. 이때, SQL 쿼리문에서 사용되는 함수명은 SQL 명을 사용하는데, 함수의 SQL 명을 별도로 정의하지 않은 경우, SQL 함수명은 메소드명과 동일하다. 다음은 위에서 정의한 저장 함수 “Cube” 의 사용예이다:

```
SELECT Cost, MyApp.Utils_Cube(Cost) As CubeCost FROM Products
```

동일 패키지(스키마)에서의 저장 함수명은 고유하여야 한다.

---

## 4. Caché SQL 참조

### 1) 집계(Aggregate) 함수

값의 집합 및 그 집계 값에 대해 수행하는 함수로서 여러 레코드의 특정 필드값들을 연산하여 단일값으로 반환한다. 지원되는 함수는 다음과 같다:

- SUM — 특정 컬럼의 합계
- AVG — 특정 컬럼의 평균값
- COUNT — 테이블이나 특정 컬럼의 행수
- MAX — 특정 컬럼의 최대값
- MIN — 특정 컬럼의 최소값
- LIST — 특정 컬럼의 값들을 콤마로 구분된 리스트로 반환
- XMLAGG — 특정 컬럼에서 사용되는 값 전체를 연결된 문자열로 반환

## 잠금

집계 함수는 보통 많은 행의 데이터를 취급하게 되는데, 합계 계산에 포함된 모든 행들에 대해서 트랜잭션 잠금을 적용하는 것은 비현실적이므로 적용되지 않는다. 그러므로 합계 계산중 다른 사용자에 의해 데이터가 수정될 가능성이 있다. 따라서, 집계 함수의 리턴값은 다른 사용자에 의해 변경된 완료되지 않은(uncommitted) 값이 적용될 수도 있다.

## 컬럼 이름

디폴트로, 집계 함수의 결과를 나타내는 컬럼명은 “Aggregate\_1”, “Aggregate\_2” 등으로 표시되며, 특정 컬럼명을 명시하려면 SELECT 문에서 함수 다음에 AS 키워드를 사용하여 정의한다.

## 2) 환경 설정

시스템 전체적으로 적용되는 Caché SQL 환경 설정은 시스템 관리 포탈 또는 제공되는 클래스 메소드나 함수를 사용하여 프로그램적으로 수행할 수 있다.

- 포탈의 메인 페이지에서 환경구성, SQL 설정, 일반 SQL 설정을 차례대로 선택한다([총] > [환경구성] > [SQL 설정] > [일반 SQL 설정]).
- 시스템 클래스 `$SYSTEM.SQL.CurrentSettings` 의 메소드들을 사용하여 동일한 설정을 수행할 수 있다.

또한 [총] > [환경구성] > [SQL 설정] 페이지의 시스템 정의 DDL 매팅 및 사용자 정의 DDL 매팅을 이용하여 시스템 및 사용자 정의 데이터 타입을 관리할 수 있다.

다음은 제공되는 Caché SQL 환경 설정 옵션들에 대한 설명이다.

### 존재하지 않는 테이블에 대한 DDL DROP 허용

“예” 또는 “아니오”로 설정. “예”인 경우, 해당 테이블명이 존재하지 않더라도 DDL을 사용하여 주어진 테이블을 삭제할 수 있다. “아니오”인 경우, 삭제를 시도하면 오류 코드가 발생된다. 디폴트는 “아니오”.

### 기존 테이블에 대해 DDL CREATE TABLE 허용

“예” 또는 “아니오”로 설정. “예”인 경우, 같은 이름의 테이블이 이미 존재하더라도 DDL을 사용하여 동일 테이블을 생성할 수 있다. “아니오”인 경우, 생성을 시도하면 오류 코드가 발생된다. 디폴트는 “아니오”. (이다)

### 키가 존재할 경우 DDL을 통해 Primary Key 생성을 허용

“예” 또는 “아니오”로 설정. “예”인 경우, 해당 테이블에 대한 기본 키(primary key)가 이미 정의되어 있더라도 DDL을 사용하여 주어진 테이블에 대한 기본 키를 생성할 수 있다. “아니오”인 경우, 생성을 시도하면 오류 코드가 발생된다. 디폴트는 “아니오”.

### 존재하지 않는 제약조건에 대한 DDL DROP 허용

“예” 또는 “아니오”로 설정. “예”인 경우, 해당 이름의 제약이 없더라도 DDL을 사용하여 주어진 제약을 삭제할 수 있다. “아니오”인 경우, 삭제를 시도하면 오류 코드가 발생된다. 디폴트는 “아니오”.

### 기존 인덱스에 대해 DDL CREATE INDEX 허용

“예” 또는 “아니오”로 설정. “예”인 경우, 동일 이름의 인덱스가 존재하더라도 DDL을 사용하여 인덱스를 생성할 수 있다. “아니오”인 경우, 생성을 시도하면 오류 코드가 발생된다. 디폴트는 “아니오”.

### 존재하지 않는 인덱스에 대해 DDL DROP 허용

“예” 또는 “아니오”로 설정. “예”인 경우, 동일 이름의 인덱스가 존재하지 않더라도 DDL을 사용하여 주어진 인덱스를 삭제할 수 있다. “아니오”인 경우, 삭제를 시도하면 오류 코드가 발생된다. 디폴트는 “아니오”.

### 외부 키가 존재할 경우 DDL ADD Foreign Key 제약조건을 허용

“예” 또는 “아니오”로 설정. “예”인 경우, 동일 이름의 외부 키가 이미 존재하더라도 DDL을 사용하여 명령을 수행할 수 있다. “아니오”인 경우, 추가를 시도하면 오류 코드가 발생된다.

디폴트는 “아니오”.

#### DDL DROP TABLE 실행 시 테이블의 데이터 삭제

“예” 또는 “아니오”로 설정. “예”인 경우, DDL DROP TABLE 이 실행될 때 해당 테이블의 데이터도 같이 삭제한다. “아니오”인 경우, 테이블 정의는 삭제되지만 데이터는 그대로 남는다. 디폴트는 “예”.

#### DDL을 통해 생성된 Primary Key가 ID Key 가 아님

“예” 또는 “아니오”로 설정. “예”인 경우, 기본 키 제약이 DDL을 통해 정의되지만 클래스 정의의 IDKey 인덱스가 되지는 않는다. “아니오”인 경우, DDL을 통하여 정의된 기본 키 제약이 클래스 정의에서의 IDKey 인덱스가 된다. 이 경우, 성능은 향상되지만 기본 키 필드들이 갱신되지 못한다는 단점이 있다. 디폴트는 “예”.

#### Lock 타임아웃(초)

SQL 문 실행시에 수행되는 Caché lock에 대한 잠금 타임아웃 시간으로, 디폴트값은 10 초이다. 허용 범위는 0-32767 초(최대 9 시간).

#### Lock 임계치

단일 트랜잭션에서 하나의 테이블에 적용되는 SQL INSERT, UPDATE, DELETE의 횟수로서 테이블 레벨(table-level) 잠금이 적용되기 이전 최대값.

예를 들어, 잠금 임계치가 1,000인 경우, 프로세스 트랜잭션이 시작하여 2,000개의 행을 삽입하였다면, 1001 번째 행을 삽입할 때 부터는 각각의 행에 Lock 이 적용되는 것이 아니라 대상 테이블 전체에 대하여 Lock 이 적용된다. 이는 Lock(잠금) 테이블이 꽉 차는 것을 방지하는데 도움이 된다. SQL 잠금 임계치는 정수값이며 디폴트값은 1000이다.

#### .INT 코드안에 주석으로서 SQL 문 유지

“예” 또는 “아니오”로 설정. “예”인 경우, 임베디드 SQL 문은 .INT 코드로 변환할 때 주석으로서 루틴안에 유지된다. 디폴트값은 “예”.

#### 구분된 식별자 지원

“예” 또는 “아니오”로 설정. “예”인 경우, 큰 따옴표 문자열("My String")은 SQL 문에서 구별 식별자로 간주된다. “아니오”인 경우, 큰 따옴표 문자열("My String")은 문자열 상수 또는 글자

문자열로 간주된다. 디풀트는 예.

### SQL 보안 가능

“예” 또는 “아니오”로 설정. “예”인 경우, 모든 Caché SQL 보안이 적용된다. 즉, 권한을 기반으로한 테이블/뷰/프로시저 보안 기능이 작동하며 이에 따라 사용자는 권한이 허용된 테이블이나 뷰에 대해서만 작업을 수행 할 수 있다. “아니오”인 경우, SQL 보안 구성은 적용되지 않으며, 따라서 사용자는 아무 제한없이 원하는 테이블이나 뷰에 접근할 수 있다. 디풀트는 “예”.

### 저장된 쿼리 – Routine Prefix

캐시된 쿼리 루틴에 사용되는 루틴 접두어로서 디풀트는 “CacheSql”이다.

### 저장된 쿼리-소스 저장

“예” 또는 “아니오”로 설정. “예”인 경우, 캐시된 쿼리 루틴의 소스 코드(.MAC 및 .INT)는 저장된다. 디풀트는 “아니오”.

### 식별자 변환 – From

DDL 식별자 변환 매핑의 “From” 목록을 제공하는 문자들로서, 최대 문자 개수는 256개이다. 디풀트 문자들은

```
~`@#$%^&*()_+-=[]₩{}|;!:","",./<>?
```

이다. 이러한 매핑은 SQL 식별자를 오브젝트 식별자로 전환할 때, SQL 식별 문자가 유효한지를 체크하고 수정한다. DDL에서 SQL 식별자를 오브젝트 식별자로 전환할 때, “From” 목록의 문자들을 “To” 문자열로 전환된다.

### 식별자 변환 – To

DDL 식별자 변환 매핑의 “To” 목록을 제공하는 문자들로서, 최대 문자 개수는 256개이다. 디풀트로 제공된 문자는 없다.

### INSERT, UPDATE, DELETE에 대한 외부 키의 참조 무결성 검사 수행

“예” 또는 “아니오”로 설정. “예”인 경우, Caché는 INSERT, UPDATE, DELETE 문의 외부 키 제약에 대한 검사를 수행한다. “아니오”的 경우, 외부 키 제약에 대한 검증을 하지 않는다. 디풀트는 “예”.

## 기본 SQL 스키마명

기본 스키마 이름. 최대 길이는 128 개 문자이다. 디폴트값은 “SQLUser” 이다.

기본 스키마명은, 주어진 SQL 문에서 부적절한 테이블명이 발견되고 **#import** 문도 정의되어 있지 않을 때 사용한다. 이 값은 SQL 스키마명과 클래스 패키지명 사이의 맵핑과는 아무 관련이 없고, 단지 기본 스키마명을 정의할 뿐이다.

기본 스키마명으로 **\_CURRENT\_USER**를 설정하면, 기본 스키마명은 현재 로그인된 프로세스의 사용자명이 된다. 프로세스가 로그인되어 있지 않으면, **SQLUser** 가 기본 스키마명이 된다. 기본 스키마명으로 **\_CURRENT\_USER/name**을 설정하면 (여기서 **name** 은 사용자가 정의한 임의의 문자열임), 기본 스키마명은 현재 로그인된 프로세스의 사용자명이 된다. 프로세스가 로그인되어 있지 않으면, 기본 스키마명으로 **name** 이 사용된다. 예를 들어, **\_CURRENT\_USER/HMO** 는 프로세스가 로그인되어 있지 않은 경우 **HMO** 가 기본 스키마명이 된다.

## SQL 문에 외래 함수 허용

“예” 또는 “아니오”로 설정. “예”인 경우, ODBC, JDBC, 동적 쿼리를 통한 SQL 문에서 외부 함수를 사용할 수 있다. 디폴트는 “아니오”.

## GETDATE, CURRENT\_TIME, CURRENT\_TIMESTAMP의 기본 시간 정밀도

SQL 스칼라 함수 **GETDATE**, **CURRENT\_TIME**, **CURRENT\_TIMESTAMP** 가 리턴하는 값의 시간 부분에 대한 기본 시간 정밀도를 말하며, 시간 값의 millisecond 부분에 허용할 소수 자리수를 정의한다. 디폴트는 0이며, 이는 시간 값에서 milliseconds 부분이 주어지지 않음을 의미한다. 설정 범위는 0~9이다.

## DISTINCT 최적화 사용

“예” 또는 “아니오”로 설정. “예”인 경우, **DISTINCT** 또는 **GROUP BY** 절을 포함하는 SQL 쿼리는 인덱스를 잘 활용하여 보다 효율적으로 실행된다. 그렇지만 이런 최적화된 쿼리에서 반환되는 값은 인덱스에 저장된 방법으로 정렬되는데, 이는 쿼리 결과가 모두 대문자로 표현될 수 있음을 의미한다. 따라서, 대소문자를 구분하는 응용프로그램에 보다 효과를 미칠 수도 있다. 디폴트는 “예”.

## TCP 유지(킵 얼라이브) 간격 (초)

TCP 연결이 유지되는 시간으로 ‘초’ 단위로 표시한 정수값이다. 유효한 값의 범위는 1~432000이다. 디폴트값은 1.

### 3) 데이터 형식

SQL 항목(컬럼 등)의 데이터 타입에 대하여 다음 내용들을 중심으로 설명한다:

- 지원되는 DDL 데이터 타입 및 매핑 테이블
- 긴 문자열, 리스트 데이터 타입, 스트림 데이터 타입 지원
- Caché ODBC / JDBC 에 사용되는 데이터 타입
- 쿼리 메타데이터 메소드와 데이터 타입 정수 코드를 사용하는 컬럼의 데이터 타입 결정.
- 사용자 정의 데이터 타입 생성
- 정의되지 않은 데이터 타입 처리
- 데이터 타입(형식) 변(전)환 함수

데이터 타입은 컬럼이 갖게되는 값의 종류를 정의한다. 일반적으로 **CREATE TABLE** 을 사용하여 필드를 정의할 때 데이터 타입을 설정한다. SQL 표현식을 정의할 때, 아래 테이블의 왼쪽 컬럼에 표시된 DDL 데이터 타입을 사용할 수 있으며, 대응하는 Caché 루틴이 생성될 때 설정된 DDL 데이터 타입은 아래의 테이블의 오른쪽 컬럼에 매핑된 Caché 데이터 타입으로 전환된다.

현재 시스템의 데이터 타입 매핑을 보려면, 시스템 관리 포탈에서 **[홈]>[환경구성] > [SQL 설정] > [시스템 정의 DDL 매핑]** 페이지로 이동한다.

추가적으로 사용자 데이터 타입을 정의할 수도 있는데, 사용자 정의 데이터 타입 매핑을 생성하거나 보려면, 시스템 관리 포탈에서 **[홈]>[환경구성]>[SQL 설정] > [사용자 정의 DDL 매핑]** 페이지로 이동한다.

DDL 데이터 타입 테이블

DDL 시스템 데이터 타입(형식)	대(상)응하는 Caché 데이터 타입(형식)
BIGINT	%Library.BigInt
BINARY	%Library.Binary(MAXLEN=1)
BINARY(%1)	%Library.Binary(MAXLEN=%1)
BINARY VARYING	%Library.Binary(MAXLEN=1)
BINARY VARYING(%1)	%Library.Binary(MAXLEN=%1)
BIT	%Library.Boolean
CHAR	%Library.String(MAXLEN=1)
CHAR(%1)	%Library.String(MAXLEN=%1)
CHAR VARYING	%Library.String(MAXLEN=1)

CHAR VARYING(%1)	<code>%Library.String(MAXLEN=%1)</code>
CHARACTER	<code>%Library.String(MAXLEN=1)</code>
CHARACTER VARYING	<code>%Library.String(MAXLEN=1)</code>
CHARACTER VARYING(%1)	<code>%Library.String(MAXLEN=%1)</code>
CHARACTER(%1)	<code>%Library.String(MAXLEN=%1)</code>
DATE	<code>%Library.Date</code>
DATETIME	<code>%Library.TimeStamp</code>
DEC	<code>%Library.Numeric</code> 64 비트 부호 있는 정수 (MAXVAL=9223372036854775807, MINVAL=-9223372036854775808, SCALE=0)
DEC(%1)	<code>%Library.Numeric</code> (MAXVAL=<  '\$\$maxval^%apiSQL(%1,0)'  >, MINVAL=<  '\$\$minval^%apiSQL(%1,0)'  >, SCALE=0). %1 의 최대 유효값은 19 이며, 19 이상의 값에 대하여 오류를 발생시키지는 않지만, 디플트값은 19이다.
DEC(%1,%2)	<code>%Library.Numeric</code> (MAXVAL=<  '\$\$maxval^%apiSQL(%1,%2)'  >, MINVAL=<  '\$\$minval^%apiSQL(%1,%2)'  >, SCALE=%2)
DECIMAL	<code>%Library.Numeric</code> 64 비트 부호 있는 정수 (MAXVAL=9223372036854775807, MINVAL=-9223372036854775808, SCALE=0)
DECIMAL(%1)	<code>%Library.Numeric</code> (MAXVAL=<  '\$\$maxval^%apiSQL(%1,0)'  >, MINVAL=<  '\$\$minval^%apiSQL(%1,0)'  >, SCALE=0) %1 의 최대 유효값은 19 이며, 19 이상의 값에 대하여 오류를 발생시키지는 않지만, 디플트값은 19이다.
DECIMAL(%1,%2)	<code>%Library.Numeric</code> (MAXVAL=<  '\$\$maxval^%apiSQL(%1,%2)'  >, MINVAL=<  '\$\$minval^%apiSQL(%1,%2)'  >, SCALE=%2)
DOUBLE	<code>%Library.Double</code> IEEE 부동 포인트 표준. 상세정보는 온라인 문서 Caché ObjectScript Reference 에서 \$DOUBLE 함수 참조.
DOUBLE PRECISION	<code>%Library.Double</code> IEEE 부동 포인트 표준. 상세정보는 온라인 문서 Caché ObjectScript Reference 에서 \$DOUBLE 함수 참조.
FLOAT	<code>%Library.Float</code>
FLOAT(%1)	<code>%Library.Float</code> (MAXVAL=<  '\$\$maxval^%apiSQL(%1)'  >, MINVAL=<  '\$\$minval^%apiSQL(%1)'  >)
IMAGE	BStream%String

INT	%Library.Integer (MAXVAL=2147483647, MINVAL=-2147483648)
INTEGER	%Library.Integer (MAXVAL=2147483647, MINVAL=-2147483648)
LONG	CStream%String
LONG BINARY	BStream%String
LONG RAW	BStream%String
LONG VARCHAR	CStream%String
LONG VARCHAR(%1)	CStream%String
LONGVARBINARY	BStream%String
LONGVARBINARY(%1)	BStream%String
LONGVARCHAR	CStream%String
LONGVARCHAR(%1)	CStream%String
MONEY	%Library.Currency (MAXVAL=922337203685477.5807, MINVAL=-922337203685477.5808, SCALE=4)
NATIONAL CHAR	%Library.String (MAXLEN=1)
NATIONAL CHAR(%1)	%Library.String (MAXLEN=%1)
NATIONAL CHAR VARYING	%Library.String (MAXLEN=1)
NATIONAL CHAR VARYING(%1)	%Library.String (MAXLEN=%1)
NATIONAL CHARACTER	%Library.String (MAXLEN=1)
NATIONAL CHARACTER(%1)	%Library.String (MAXLEN=%1)
NATIONAL CHARACTER VARYING	%Library.String (MAXLEN=1)
NATIONAL CHARACTER VARYING(%1)	%Library.String (MAXLEN=%1)
NATIONAL VARCHAR	%Library.String (MAXLEN=1)
NATIONAL VARCHAR(%1)	%Library.String (MAXLEN=%1)
NCHAR	%Library.String (MAXLEN=1)
NCHAR(%1)	%Library.String (MAXLEN=%1)
NTEXT	CStream%String
NUMBER	%Library.Numeric 64 비트 부호 있는 정수 (MAXVAL=9223372036854775807,

	MINVAL=-9223372036854775808, SCALE=0)
NUMBER(%1)	%Library.Numeric (MAXVAL=<  '\$\$maxval^%apiSQL(%1)' >, MINVAL=<  '\$\$minval^%apiSQL(%1)' >, SCALE=0) %1 의 최대 유효값은 19 이며, 19 이상의 값에 대하여는 오류를 발생시키지는 않지만, 디폴트값은 19이다.
NUMBER(%1,%2)	%Library.Numeric (MAXVAL=<  '\$\$maxval^%apiSQL(%1,%2)' >, MINVAL=<  '\$\$minval^%apiSQL(%1,%2)' >, SCALE=%2)
NUMERIC	%Library.Numeric 64 비트 부호 있는 정수 (A 6r.) (MAXVAL=9223372036854775807, MINVAL=-9223372036854775808, SCALE=0)
NUMERIC(%1)	%Library.Numeric (MAXVAL=<  '\$\$maxval^%apiSQL(%1,0)' >, MINVAL=<  '\$\$minval^%apiSQL(%1,0)' >, SCALE=0) %1 의 최대 유효값은 19 이며, 19 이상의 값에 대하여는 오류를 발생시키지는 않지만, 디폴트값은 19이다.
NUMERIC(%1,%2)	%Library.Numeric (MAXVAL=<  '\$\$maxval^%apiSQL(%1,%2)' >, MINVAL=<  '\$\$minval^%apiSQL(%1,%2)' >, SCALE=%2)
NVARCHAR	%Library.String (MAXLEN=1)
NVARCHAR(%1)	%Library.String (MAXLEN=%1)
NVARCHAR(%1,%2)	%Library.String (MAXLEN=%1)
RAW(%1)	%Library.Binary (MAXLEN=%1)
REAL	%Library.Float
SERIAL	%Library.Counter
SMALLDATETIME	%Library.TimeStamp
SMALLINT	%Library.SmallInt
SMALLMONEY	%Library.Currency
SYSNAME	%Library.String (MAXLEN=128)
TEXT	CStream%String
TIME	%Library.Time
TIMESTAMP	%Library.TimeStamp
TINYINT	%Library.TinyInt
VARBINARY	%Library.Binary (MAXLEN=1)
VARBINARY(%1)	%Library.Binary (MAXLEN=%1)
VARCHAR	%Library.String (MAXLEN=1)
VARCHAR(%1)	%Library.String (MAXLEN=%1)

VARCHAR(%1,%2)	%Library.String (MAXLEN=%1)
VARCHAR2(%1)	%Library.String (MAXLEN=%1)

## SQL 시스템 데이터 타입 매핑

위 테이블에 기술된 DDL 과 Caché 데이터 타입 표현식은 **SQL.SystemDataTypes** 에 구성된 기본 매핑이다. 제공된 시스템 데이터 타입과 사용자 데이터 타입에 대해서는 별도의 매핑 테이블이 존재한다.

현재 정의된 데이터 타입 매핑을 보고 수정하려면, 시스템 관리 포탈에서 [총] > [환경구성] > [SQL 설정] > [시스템 정의 DDL 매핑] 페이지로 이동한다.

## DDL 데이터 타입 매핑 이해

데이터 타입을 DDL에서 Caché로 매핑할 때, 정규 파라미터와 함수 파라미터는 다음의 규칙을 따른다:

- 정규 파라미터: DDL 과 Caché 데이터 타입에서 %# 형식으로 식별된다. 예를 들어, DDL 데이터 타입

VARCHAR(%1)

은 Caché 타입

%String(MAXLEN=%1)

으로 매핑한다. 따라서, DDL 데이터 타입

VARCHAR(10)

은 다음과 같이 Caché 에 대응된다.

%String(MAXLEN=10)

- 함수 파라미터: DDL 데이터 타입 파라미터가 Caché 데이터 타입으로 최종 변환되기전 중간 단계의 변환이 이루어져야 하는 경우에 사용된다. 이러한 경우로는, DDL 데이터 타입의 숫자 정밀도와 스케일 파라미터가, Caché 데이터 타입의 MAXVAL, MINVAL 및 SCALE 파라미터로 변형되는 경우를 들 수 있다. 예를 들면, DDL 데이터 타입

DECIMAL(%1,%2)

은 Caché 데이터 타입

```
%Numeric(MAXVAL=<| '$$maxval^%apiSQL(%1,%2)'|>,
        MINVAL=<| '$$minval^%apiSQL(%1,%2)'|>, SCALE=%2)
```

으로 매핑한다.

DDL 데이터 타입 **DECIMAL**은 파라미터 Precision (%1)과 Scale (%2)를 취하지만, Caché 데이터 타입 **%Numeric**은 Precision 파라미터가 없다. 따라서, **DECIMAL**을 **%Numeric**으로 전환하려면, Precision 파라미터는 적절한 **%Numeric** 파라미터로 전환되어야 하며, 이런 경우, Caché 함수 *format*, *maxval*, *minval*을 **DECIMAL** 파라미터에 적용한다. 위의 문법에서 사용한 <|'xxx'|> 표현식은 DDL 처리기로 하여금 파라미터를 주어진 값으로 교체하는 함수를 호출하도록 한다. 그러면 <|'xxx'|> 표현식은 호출된 함수가 리턴하는 값으로 대체된다.

다음의 정밀도 4 자리 숫자와 스케일 2의 값을 가지는 DDL DECIMAL 데이터 타입은

```
DECIMAL(4,2)
```

Caché 타입

```
%Numeric(MAXVAL=<| '$$maxval^%apiSQL(4,2)'|>,
        MINVAL=<| '$$minval^%apiSQL(4,2)'|>, SCALE=2)
```

으로 매핑되며, 이 값은 다음과 같다:

```
%Numeric(MAXVAL=99.99,MINVAL=-99.99,SCALE=2).
```

## 긴 문자열

Caché는 긴 문자열을 지원한다. 긴 문자열은 32,749 문자(64K 바이트) 이상의 데이터를 갖는 문자열을 의미하며, 이러한 스트링은 **%CStream%String** (또는 **%Library.GlobalCharacterStream**) 데이터 타입으로 정의하여야 한다.

일반적으로 **%Library.String** 데이터 타입에 16,374 문자 이상의 MAXLEN을 지정할 수는 있지만, ODBC나 JDBC로 MAXLEN이 16,374 문자 이상인 **%Library.String** 데이터에 액세스하게 되면, 처음 16,374 개의 문자만이 리턴된다. 따라서 16,374 이상의 문자를 갖는 데이터는 반드시 스트림 데이터 타입을 사용하여야 한다.

긴 문자열 지원은 시스템 관리 포탈 또는 Caché ObjectScript \$ZUTIL(69,69) 함수를 사용하여 설정할 수 있다. 시스템 관리 포탈에서 **[홈] > [환경구성] > [메모리 및 시작]** 페이지로 이동한 후 ‘긴 문자열 허용’을 선택하고 저장 버튼을 누른다. 디폴트는 긴 문자열을 허용하지 않은

상태이다.

## 리스트 구조

Caché 는 데이터 타입 %Library.List 의 리스트 구조를 제공한다. 이는 압축된 바이너리 형식으로서, Caché SQL 의 어떠한 고유 데이터 타입과도 매핑되지 않는다. 내부적인 표현으로서는 32,749 의 MAXLEN 를 갖는 VARBINARY 데이터 타입에 대응한다.

이런 이유로, 동적 SQL은 WHERE 절 비교에서 %List 데이터를 사용할 수 없으며, %List 타입의 프로퍼티 값을 설정하기 위해 INSERT, UPDATE를 사용할 수 없다. 동적 SQL 에서는 리스트 구조의 데이터 타입을 VARCHAR로 반환한다.

ODBC 또는 JDBC 클라이언트에서는, %List 데이터는 LogicalToOdbc 변환을 사용하여, VARCHAR 문자열 데이터로 전환되며, 리스트의 데이터는 각 요소를 콤마로 구분하여 문자열로 전환된다. 이렇게 전환된 형식의 문자열 데이터는 WHERE 절을 비롯하여 INSERT 및 UPDATE 문에서 사용될 수 있다.

Caché SQL 는 8 개의 리스트 관련 함수를 지원한다: \$LIST, \$LISTBUILD, \$LISTDATA, \$LISTFIND, \$LISTFROMSTRING, \$LISTGET, \$LISTLENGTH, \$LISTTOSTRING.

또한 Caché ObjectScript 은 세 개의 리스트 함수를 추가로 제공한다:

- \$LISTVALID: 주어진 데이터가 리스트 형식인지 확인
- \$LISTSAME: 두 개의 리스트 데이터 비교
- \$LISTNEXT: 주어진 리스트의 요소들을 순차적으로 검색

## 스트림 데이터 타입

스트림 데이터 타입의 필드는 SQL 스칼라, 집계, 단항 함수에 대한 인자로 사용할 수 없다. 하나의 예외는 %INTERNAL 함수인데, 스트림 필드를 받아들이기는 하나 이 필드에 대하여 아무런 연산도 수행하지 않는다.

스트림 데이터 타입을 WHERE 절이나 인덱스, 또는 INSERT 및 UPDATE 작업에 사용하는 것도 제한된다.

## SERIAL 데이터 타입

SERIAL 데이터 타입의 필드는 사용자 지정 양의 정수 값을 갖거나, Caché 로부터 순차적인 양의 정수 값을 지정 받을 수 있다. INSERT 작업은 SERIAL 필드에 다음 중 하나의 값을 설정한다:

- 값이 없거나, 0 또는 숫자가 아닌 값: Caché 는 설정된 값을 무시하고, 대신 이 필드의 현재 시리얼 카운터 값을 1 증가시킨 후, 그 증가시킨 값을 필드에 삽입한다.

- 양의 정수 값: Caché는 사용자 지정 값을 필드에 삽입하고, 이 필드의 시리얼 카운터 값을 사용자 지정 값으로 바꾼다.

따라서 SERIAL 필드는 하나의 연속적인 증가 정수 값을 가지게 된다. 그렇지만, 그 값들이 반드시 연속적이거나 유일한 값일 필요는 없다. 예를 들어, 1, 2, 3, 17, 18, 25, 25, 26, 27과 같은 값들은 SERIAL 필드의 값으로 유효한 값이다. 순차 정수는 Caché에서 생성하거나 또는 사용자가 지정할 수 있으나, 비 순차 정수는 사용자가 지정한다. SERIAL 필드값이 유일하기를 원한다면, 필드에 대하여 **UNIQUE** 제약을 적용한다

**UPDATE** 작업은 SERIAL 필드의 현재값이 없거나(NULL), 또는 값이 0인 경우에만 그 값을 바꿀 수 있으면, 그 외의 경우에는 SQLCODE -105 오류 코드가 발생한다. 테이블에서의 SERIAL 필드의 개수에 대한 제한은 없다.

#### Caché ODBC / JDBC에 표현되는 DDL 데이터 타입

Caché ODBC는 DDL 데이터 타입의 부분집합으로 표현되며 외부 데이터 타입들을 이 데이터 타입 집합으로 매핑하는데 그 역매핑은 성립하지 않는다. 예를 들어, **CREATE TABLE mytable (f1 BINARY)** 문은 **mytable (f1 VARBINARY)**로서 ODBC에 전환되는 Caché 클래스를 생성한다. Caché 리스트 데이터 타입은 ODBC에 VARCHAR 타입으로 전환된다.

ODBC는 다음의 데이터 타입들을 표현한다: **BIT, DATE, DOUBLE, INTEGER, LONGVARBINARY, LONGVARCHAR, NUMERIC, SMALLINT, TIME, TIMESTAMP, TINYINT, VARBINARY, VARCHAR**.

이러한 ODBC/JDBC 데이터 타입 값이 Caché SQL로 매핑될 때, DOUBEB 데이터는 **\$DOUBLE**을, NUMERIC 데이터는 **\$DECIMAL**을 사용하여 변환된다.

#### 데이터 타입을 리턴하는 메타 데이터 쿼리

특정 컬럼의 데이터 타입 등을 포함한 쿼리의 메타 데이터를 리턴하기 위해 동적 SQL을 다음과 같이 사용할 수 있다. 이 쿼리는 Sample.Person의 각 컬럼에 대한 컬럼명과 데이터 타입 코드를 반환한다:

```
SET myquery="SELECT * FROM Sample.Person"
SET rset=##class(%ResultSet).%New("%DynamicQuery:SQL")
SET sc=rset.Prepare(myquery)
SET x=rset.GetColumnCount()
WHILE x>0 {
  WRITE !,x," ",rset.GetColumnName(x)," ",rset.GetColumnType(x)
  SET x=x-1 }
WRITE !,"end of columns"
```

## 데이터 타입 코드

컬럼의 메타 데이터에서 해당 컬럼의 데이터 타입은 다음과 같은 두 종류의 코드값 중 하나로 반환된다.

- 클라이언트 데이터 타입 코드는 %Library.ResultSet.GetColumnType() 메소드에 의해 리턴된다. 리턴되는 코드(이) 값들은 다음과 같다:

1	BINARY
2	DATE
3	DOUBLE
4	HANDLE
5	INTEGER
6	LIST
7	LONGVARCHAR
8	TIME
9	TIMESTAMP
10	VARCHAR
11	STATUS
12	BINARYSTREAM
13	CHARACTERSTREAM
14	NUMERIC
15	CURRENCY
16	BOOLEAN
17	OID
18	BIGINT
19	FDATE
20	FTIMESTAMP

- xDBC 데이터 타입 코드는 ODBC와 JDBC에 의해 사용되며 코드값은 %ResultSet.SQL.%GetMetaData() 메소드에 의해 반환된다. ODBC와 JDBC 코드는, 시간과 날짜 데이터 타입의 경우만 틀리고 그 외의 코드값들은 동일하다. ODBC와 JDBC에서의 코드값 목록은 다음과 같다.

ODBC	JDBC	데이터 형식
-7	-7	BIT
-6	-6	TINYINT
-5	-5	BIGINT
-4	-4	LONGVARBINARY
-3	-3	VARBINARY
-1	-1	LONGVARCHAR
0	0	Unknown type
1	1	CHAR
2	2	NUMERIC

3	3	DECIMAL
4	4	INTEGER
5	5	SMALLINT
6	6	FLOAT
7	7	REAL
8	8	DOUBLE
9	91	DATE
10	92	TIME
11	93	TIMESTAMP
12	12	VARCHAR

## 사용자 정의 DDL 데이터 타입 만들기

시스템 데이터 타입 파라미터 값에 대한 데이터 타입 매핑을 재정의하거나 사용자 데이터 타입을 새로 정의함으로써 데이터 타입을 변경할 수 있다.

현재 정의되어 있는 사용자 데이터 타입 매핑을 보고, 수정, 또는 추가하기 위해서는 시스템 관리 포탈에서 **[홈] > [환경구] > [SQL 설정] > [사용자 정의 DDL 매핑]** 페이지로 이동한다. 사용자 데이터 타입을 추가하려면, “새 사용자 정의 DDL 매핑 생성” 메뉴를 선택한다. 나타난 페이지에서 이름 및 Datatype 을 입력한다. 예를 들어, 이름에는 MyString100(MAXLEN=100) 을, Datatype 에는 VARCHAR(100) 과 같은 값을 입력한다. 저장 버튼을 눌려 사용자 정의 DDL 데이터 타입 목록에 추가한다.

다음은 사용자 정의 DDL 데이터 타입을 직접 수정할 수 있는 두 루틴을 소개한다:

- **maxval^%apiSQL()**: Caché 수치 데이터 타입들의 최대 유효값을 리턴한다:

```
maxval^%apiSQL(precision,scale)
```

- **minval^%apiSQL()**: Caché 수치 데이터 타입의 최소 유효값을 리턴한다:

```
minval^%apiSQL(precision,scale)
```

DDL 데이터 타입을 스트림의 컬렉션 타입을 갖는 Caché 프로퍼티에 매핑하려면, 문자 스트림 데이터에 대해서는 **CStream%String** (또는 **%Library.GlobalCharacterStream**)을, 바이너리 스트림 데이터에 대해서는 **BStream%String** (또는 **%Library.GlobalBinaryStream**)을 설정한다.

### DDL 매핑이 없으면 통과하기

만약, DDL 이 **UserDataTypes** 테이블의 DDL 데이터 타입 컬럼에 정의되어 있지 않은 데이터 타입을 감지 했다면, 다음으로 **SystemDataTypes** 테이블을 검색한다. 두 테이블 모두에서 주어진 데이터 타입에 대한 매핑을 찾지 못했다면, 해당 데이터 타입에 대한 변환은 수행되지 않으며 DDL에서 지정된 클래스 정의로 직접 연결된다.

예를 들어(서), 다음의 DDL 필드 정의에서,

```
CREATE TABLE TestTable (
    Field1 %String,
    Field2 %String(MAXLEN=45)
)
```

DDL이 **UserDataTypes** 와 **SystemDataTypes** 테이블에서 **%String**, **%String(MAXLEN=%1)**, 또는 **%String(MAXLEN=45)** 에 대한 매핑을 찾지 못하면, **%String** 과 **%String(MAXLEN=45)** 타입은 직접 지정된 클래스 정의로 연결된다.

### 데이터 형식 전환

**CAST** 또는 **CONVERT** 함수를 사용하여 데이터의 데이터 타입을 전환한다.

**CAST** 는 문자열, 수치 데이터 타입 뿐만 아니라 DATE, TIME, TIMESTAMP 에 대한 전환도 지원한다.

**CONVERT** 는 두 가지의 구문 형식을 갖는데, 두 가지 모두 DATE, TIME, TIMESTAMP 데이터 타입 뿐만 아니라 그 외의 데이터 타입간의 전환도 지원한다.

크기가 설정되어 있지 않은 **VARCHAR** 데이터 타입은 MAXLEN 1 인 문자로 매핑되는데, **CAST** 또는 **CONVERT** 를 사용하여 주어진 값을 **VARCHAR** 로 전환하는 경우에는, 디폴트 매핑 사이즈가 30 문자이며, 이는 Caché 이외의 제품들과의 호환을 위한 것이다.

## 4) 날짜와 시간 구조 (Date and Time Constructs)

주어진 문자열을 날짜, 시간, 타임 스템프로 형식화한다.

```
{d 'yyyy-mm-dd'}
{t 'hh:mm:ss'}
{ts 'yyyy-mm-dd hh:mm:ss.fff'}
```

이 형식화 작업은 ODBC 날짜와 시간 형식을 대응하는 Caché 내부 형식인 \$HOROLOG 의 날짜, 시간, 타임스탬프로 전환하며 이 때, 주어진 데이터의 형식 및 범위를 체크한다.

타임스탬프 문자열이 주어졌을 경우에는,

- **{d 'string'}** 날짜 구조를 검증하고 날짜 부분을 표시한다. 이 때, 시간 부분은 무시되며, 날짜에 대한 검증이 실패하면 내부 값으로 0 이 설정된다. Display 모드에서 0 의 날짜 값은 '12/31/1840' 로 나타나며, ODBC 모드에서는 '1840-12-31' 로 표현된다.
- **{t 'string'}** 주어진 값의 시간값을 검증하며 이 때 초 이하의 값은 무시한다. 시간 검증에 실패한 경우, 내부 값으로 0 이 설정되며, Display 모드 및 ODBC 모드에서 모두 '00:00:00' 으로 표현된다.
- **{ts 'string'}** 타임스탬프 구조는 초 이하의 값을 포함하여 주어진 모든 날짜와 시간 문자열을 전환한다. 날짜 값만이 주어졌을 경우, ODBC 형식으로 전환되며 시간은 00:00:00 으로 나타낸다. 시간 값만이 주어지거나 부적절한 날짜 값이 주어지면, 변경 없이 주어진 문자열을 그대로 반환한다.

다음은 주어진 값을 Display 모드에서 변환하는 예로서, **LeapDate** 필드는 '02/29/2004' 로, **NonLeapDate** 는 윤년 검증을 실패하여 0 을 리턴하며 따라서 0 의 날짜 '12/31/1840' 으로 표시된다. **TSGoodDate** 필드는 날짜값만이 주어졌기 때문에 ODBC 타임스탬프 형식으로 '2003-02-28 00:00:00' 를 리턴하며, **TSBadDate** 필드의 경우, 날짜 검증에 실패했지만 타임스탬프 구조이므로 주어진 문자열 '2003-02-29' 를 그대로 반환한다:

```
SELECT DISTINCT {d '2004-02-29'} AS LeapDate,
{d '2003-02-29'} AS NonLeapDate,
{ts '2003-02-28'} AS TSGoodDate,
{ts '2003-02-29'} AS TSBadDate
FROM Sample.Person
```

## 5) 디풀트 사용자명과 패스워드

Caché 데이터베이스에 액세스하기 위해서는 로그인 계정이 필요한데, Caché 에서 제공하는 디풀트 사용자명 및 패스워드는 각각 '\_SYSTEM" 과 "SYS" 이다.

## 6) 필드 제약

Caché SQL 필드는 다음과 같은 제약 조건을 갖을 수 있다:

- **NOT NULL:** 해당 필드의 필드값은 NULL이 아닌 값으로 반드시 설정되어야 한다(빈 문자열도 가능).
- **UNIQUE:** 해당 필드의 필드값은 필드가 정의된 테이블의 모든 레코드에서 유일한 값이어야 한다(빈 문자열 가능). 그러나 NULL 필드값으로 레코드를 생성할 수는 있다.
- **DEFAULT:** 해당 필드에 대한 디폴트값을 반드시 정의하여야 하는데, 그 값은 NULL 이거나 데이터 타입에 따라 적당한 값을 설정한다.
- **UNIQUE NOT NULL:** 해당 필드의 필드값은 필드가 정의된 테이블의 모든 레코드에서 NULL 이 아닌 유일한 값이어야 한다(빈 문자열 가능). 이 필드는 기본 키(Primary Key)로 사용될 수 있다.
- **DEFAULT NOT NULL:** 해당 필드의 필드값은 NULL이 아닌 값으로 디폴트 값이 정의되어져야 하며, 그렇지 않은 경우, Caché 자체에서 필드의 데이터 타입에 기반한 디폴트 값을 제공할 수도 있다(빈 문자열 가능).
- **UNIQUE DEFAULT:** 권장 안함 — 해당 필드의 필드값은 필드가 정의된 테이블의 모든 레코드에서 유일한 값이 설정되거나, 필드의 데이터 타입에 따라 Caché 자체에서 디폴트 값을 제공할 수도 있다(빈 문자열 가능). 디폴트 값으로는 NULL, 빈 문자열, 또는 데이터 타입에 적당한 값이면 된다. 유일하게 생성되는 값 (예를 들면, CURRENT\_TIMESTAMP) 또는 단 한 번만 사용되는 값만을 사용한다.
- **UNIQUE DEFAULT NOT NULL:** 권장 안함 — 해당 필드의 필드값은 필드가 정의된 테이블의 모든 레코드에서 유일한 값이 설정되거나, 필드의 데이터 타입에 따라 Caché 자체에서 디폴트 값을 제공할 수도 있다(빈 문자열 가능). 디폴트 값으로는 빈 문자열 또는 데이터 타입에 적당한 값이면 되지만 NULL은 허용되지 않는다. 유일하게 생성되는 값 (예를 들면, CURRENT\_TIMESTAMP) 또는 단 한 번만 사용되는 값만을 사용한다. 기본 키(Primary Key)로 사용될 수 있다.
- **IDENTITY:** Caché 가 해당 필드가 정의된 테이블의 모든 레코드에서 이 필드에 대한 고유하고, 시스템 정의의, 변경 불가능한 정수 값을 제공하며, 이 때 다른 필드 제약 조건 키워드는 무시된다. 기본 키(Primary Key)로 사용될 수 있다.

데이터 값은 해당 필드의 데이터 타입에 맞는 형식이어야 한다. 빈 문자열은 수치 타입의 필드에는 사용할 수 없다.

## 7) 예약어

### SQL 예약어 목록

```
%AFTERHAVING | %ALPHAUP | %ALTER | %ALTER_USER | %BEGTRANS | %CHECKPRIV
| %CREATE_ROLE | %CREATE_USER | %DBUGFULL | %DELEDATA | %DESCRIPTION
| %DROP_ANY_ROLE | %DROP_USER | %EXACT | %EXTERNAL | %FILE | %FLATTEN
```

```
| %FOREACH | %FULL | %GRANT_ANY_PRIVILEGE | %GRANT_ANY_ROLE | %IGNOREINDICES  
| %INLIST | %INORDER | %INTERNAL | %INTEXT | %INTRANS | %INTRANSACTION | %MERGE  
| %MCODE | %NOCHECK | %NODEDATA | %NOFLATTEN | %NOINDEX | %NOLOCK  
| %NOMERGE | %NOSVSO | %NOTOPOPT | %NOTRIGGER | %NUMROWS | %ODBCOUT  
| %ROUTINE | %ROWCOUNT | %STARTSWITH | %STRING | %THRESHOLD | %UPPER |  
ABSOLUTE | ADD | ALL | ALLOCATE | ALTER | AND | ANY | ARE | AS | ASC | ASSERTION |  
AT | AUTHORIZATION | AVG | BEGIN | BETWEEN | BIT | BIT_LENGTH | BOTH | BY | CASCADE  
| CASE | CAST | CATALOG | CHAR | CHARACTER | CHARACTER_LENGTH | CHAR_LENGTH |  
CHECK | CLOSE | COALESCE | COLLATE | COMMIT | CONNECT | CONNECTION |  
CONSTRAINT | CONSTRAINTS | CONTINUE | CONVERT | CORRESPONDING | COUNT | CREATE  
| CROSS | CURRENT | CURRENT_DATE | CURRENT_TIME | CURRENT_TIMESTAMP |  
CURRENT_USER | CURSOR | DATE | DEALLOCATE | DEC | DECIMAL | DECLARE | DEFAULT  
| DEFERRABLE | DEFERRED | DELETE | DESC | DESCRIBE | DESCRIPTOR | DIAGNOSTICS |  
DISCONNECT | DISTINCT | DOMAIN | DOUBLE | DROP | ELSE | END | ENDEXEC | ESCAPE |  
EXCEPT | EXCEPTION | EXEC | EXECUTE | EXISTS | EXTERNAL | EXTRACT | FALSE | FETCH |  
FILE | FIRST | FLOAT | FOR | FOREIGN | FOUND | FROM | FULL | GET | GLOBAL | GO |  
GOTO | GRANT | GROUP | HAVING | HOUR | IDENTITY | IMMEDIATE | IN | INDICATOR |  
INITIALLY | INNER | INPUT | INSENSITIVE | INSERT | INT | INTEGER | INTERSECT | INTERVAL |  
INTO | IS | ISOLATION | JOIN | KEY | LANGUAGE | LAST | LEADING | LEFT | LEVEL | LIKE |  
LOCAL | LOWER | MATCH | MAX | MIN | MINUTE | MODULE | MONTH | NAMES | NATIONAL |  
NATURAL | NCHAR | NEXT | NO | NOT | NULL | NULLIF | NUMERIC | OCTET_LENGTH | OF |  
ON | ONLY | OPEN | OPTION | OR | ORDER | OUTER | OUTPUT | OVERLAPS | PAD | PARTIAL  
| PREPARE | PRESERVE | PRIMARY | PRIOR | PRIVILEGES | PROCEDURE | PUBLIC | READ |  
REAL | REFERENCES | RELATIVE | RESTRICT | REVOKE | RIGHT | ROLE | ROLLBACK | ROWS  
| SCHEMA | SCROLL | SECOND | SECTION | SELECT | SESSION_USER | SET | SMALLINT |  
SOME | SPACE | SQLERROR | SQLSTATE | STATISTICS | SUBSTRING | SUM | SYSDATE |  
SYSTEM_USER | TABLE | TEMPORARY | THEN | TIME | TIMESTAMP | TIMEZONE_HOUR |  
TIMEZONE_MINUTE | TO | TOP | TRAILING | TRANSACTION | TRANSLATE | TRANSLATION |  
TRIM | TRUE | UNION | UNIQUE | UPDATE | UPPER | USER | USING | VALUES | VARCHAR |  
VARYING | WHEN | WHENEVER | WHERE | WITH | WORK | WRITE
```

SQL 는 특정 단어들을 예약어로 등록하여 사용하고 있는데 이러한 예약어들은 다음의 경우를 제외하고는 SQL 식별자(테이블이나 컬럼 또는 다른 개체의 이름)로 사용할 수 없다:

- 큰 따옴표로 구별된 경우 (예, "word")

- ‘구분된 식별자 지원’이 설정된 경우([홀] > [환경구성] > [SQL 설정] > [일반 SQL 설정])

위의 예약어 목록은 이런 경우에 기반한 단어들만을 포함한 것으로 모든 SQL 예약어를 포함하고 있지는 않다.

주어진 단어가 SQL 예약어인지를 여부는 \$SYSTEM.SQL 클래스 메소드 `IsReservedWord()`를 사용하여 확인할 수 있다. 이 메소드는 그 결과를 Boolean 값으로 리턴한다.

```
WRITE !,"Reserved?: ",$SYSTEM.SQL.IsReservedWord("VARCHAR")
WRITE !,"Reserved?: ",$SYSTEM.SQL.IsReservedWord("varchar")
WRITE !,"Reserved?: ",$SYSTEM.SQL.IsReservedWord("VarChar")
WRITE !,"Reserved?: ",$SYSTEM.SQL.IsReservedWord("FRED")
```

이 메소드는 ODBC 또는 JDBC를 통해 저장 프로시저로 호출 가능하다:

```
%SYSTEM.SQL_IsReservedWord("nnnn").
```

## 8) 문자열 제어

Caché SQL에서는 다음과 같은 방법으로 문자열 데이터를 제어한다:

- 문자열은 길이, 문자 위치 또는 부분 문자열 값으로 제어 가능하다.
- 지정된 구별 문자 또는 구별 문자열로 제어 가능하다.
- 패턴 매치나 단어 식별 검색으로 제어 가능하다.
- 특별히 인코딩된 문자열, 리스트, 는 특정 구별 문자를 사용하지 않고도 구별된 부분 문자열들을 포함하고 있으며, 표준 문자열과는 호환되지 않는다. 다양한 \$LIST 함수들을 사용하여 리스트 데이터를 제어한다. 유일한 예외는 \$LISTGET과 \$LIST 함수인데, 이들 함수는 리스트 문자열을 입력값으로 받아 단일 값인 표준 문자열을 리턴한다.

Caché SQL는 문자열 함수, 문자열 조건부 표현식, 및 문자열 연산자를 지원한다.

문자열의 각 문자들은 대문자 또는 소문자로 변환 가능하며, 대소문자 전환은 조합을 결정하는데 자주 사용된다. Caché SQL은 다양한 대소문자 및 조합 관련 함수와 연산자를 제공한다.

주어진 문자열 인수가 수치적인 값을 의미하는 경우, 대부분의 Caché SQL 함수는 다음과 같이 문자열을 숫자로 전환하는 작업을 수행한다:

- 숫자가 아닌 문자열은 0으로 전환
- 숫자 문자열은 정규 숫자로 전환
- 숫자와 문자가 섞여있는 문자열은 숫자가 아닌 첫번째 문자 이전까지의 부분을 정규 숫자로 전환.

## 문자열 연결

- **CONCAT:** 두 개의 문자열을 연결하여 하나의 문자열을 리턴한다.
- **STRING:** 두 개의 이상의 문자열을 연결하여 하나의 문자열을 리턴한다.
- **XMLEGG:** 컬럼의 모든 값을 연결하여 하나의 문자열을 리턴한다.
- **LIST:** 콤마 구별자를 포함하여 한 컬럼의 모든 값을 연결하여 하나의 문자열을 리턴한다.

## 문자열 길이

- **CHARACTER\_LENGTH** 와 **CHAR\_LENGTH:** 문자열 끝의 공백을 포함하여, 주어진 문자열의 총 문자 개수를 리턴한다. NULL 값은 NULL 을 반환한다.
- **LENGTH:** 문자열 끝의 공백을 제외한 문자의 개수를 리턴한다. NULL 값은 NULL 을 반환한다.
- **\$LENGTH:** 문자열 끝의 공백을 포함하여 주어진 문자열의 총 문자 개수를 리턴한다. NULL 값은 0 이 반환된다.

## 자르기와 트리밍

다음의 함수들은 주어진 문자열을 자르거나 정돈(trim) 한다. 자르기(Truncation)는 주어진 길이 이상의 모든 문자들은 제거함으로써 문자열의 길이를 제한한다. 트리밍(Trimming)은 문자열 앞뒤의 공백을 삭제한다:

- 자르기(Truncation): CONVERT, %SQLSTRING, %SQLUPPER, %STRING
- 트리밍: TRIM, LTRIM, RTRIM

## 부분 문자열 검색

다음 함수들을 사용하여 문자열에서의 부분 문자열 검색을 수행한다:

- **CHARINDEX:** 주어진 부분 문자열 검색, 처음으로 일치하는 부분을 찾아 부분 문자열의 시작 위치를 리턴한다. 검색 시작위치를 설정할 수 있으며, 디폴트 시작 위치는 대상 문자열의 처음이다.
- **\$FIND:** 주어진 부분 문자열 검색, 처음 일치하는 부분을 찾아 부분 문자열의 마지막 위치를 리턴한다. 검색 시작위치를 설정할 수 있으며, 디폴트 시작 위치는 대상 문자열의 처음이다.
- **\$EXTRACT:** 문자열 위치로 검색, 주어진 시작 위치, 또는 시작 및 종료 위치에 의해 결정된 부분 문자열을 리턴한다. 문자열의 처음에서부터 검색한다.
- **SUBSTRING:** 문자열 위치로 검색, 주어진 시작 위치, 또는 시작위치 및 길이에 의해 결정된 부분 문자열을 리턴한다. 문자열의 처음에서부터 검색한다.
- **SUBSTR:** 문자열 위치로 검색, 시작 위치, 또는 시작위치 및 길이로 결정된 부분 문자열을

리턴한다. 문자열의 시작 또는 마지막에서부터 검색한다.

- **\$PIECE**: 구분자로 검색, 처음으로 발견된 구별 부분 문자열을 리턴한다. 시작 위치는 주어질 수도 있으며, 그렇지 않은 경우 디폴트로 문자열의 처음부터 시작한다.
- **\$LENGTH**: 구분자로 검색, 구분된 부분 문자열의 개수를 리턴한다. 문자열의 처음에서부터 검색한다.
- **\$LIST**: 리스트 문자열에 포함된 부분 문자열의 개수를 검색. 부분 문자열 카운트로 부분 문자열을 검색하고 찾은 부분 문자열을 리턴한다. 문자열의 처음에서부터 검색한다.

포함 연산자 ( [ ] ) 를 사용하여 부분 문자열을 검색할 수도 있다. 또한, %STARTSWITH 비교 연산자는 주어진 문자(들)로 시작하는 부분 문자열을 찾아낸다.

### 부분 문자열 검색과 대체

다음 함수는 문자열에서 부분 문자열을 검색하고 찾아낸 부분을 다른 문자열로 변경하는 작업을 수행한다:

- **REPLACE**: 문자열 값을 검색, 검색된 부분을 주어진 문자열로 대체한다. 문자열의 처음에서부터 검색한다.
- **STUFF**: 문자열 위치와 길이로 검색, 검색된 부분을 주어진 문자열로 대체한다. 문자열의 처음에서부터 검색한다.

### 문자 타입 및 단어 인식(word-aware) 비교

%PATTERN 비교 연산자는 문자열을 주어진 문자 타입의 패턴과 비교한다.

%CONTAINS 와 %CONTAINSTERM 비교 연산자는 주어진 단어나 어구에 대하여 단어별 검색을 수행한다.

# 제 6 장 다차원 배열, 글로벌의 이해

1. 특징
2. 글로벌의 구조
  - 1) 글로벌의 로직 구조
  - 2) 글로벌의 물리적 구조
  - 3) 글로벌 참조하기
3. 글로벌의 다차원 저장 방법
  - 1) 글로벌에서 데이터 저장
  - 2) 글로벌 노드 삭제하기
  - 3) 글로벌 노드 존재 여부 테스트
  - 4) 글로벌 노드 값 검색
  - 5) 글로벌 노드 데이터 찾아가기
  - 6) 글로벌 데이터 복사하기
  - 7) 글로벌 내보내기
  - 8) 글로벌 가져오기
  - 9) 글로벌 공유 카운터 관리
  - 10) 임시 글로벌 사용하기
  - 11) 글로벌 데이터 정렬
  - 12) 글로벌에 대한 간접 명령 사용
  - 13) 트랜잭션 관리
  - 14) 병행 처리 관리
  - 15) 최근 참조 글로벌 확인

다차원 저장 엔진(Multi-Dimensional Storage Engine)은 Caché 의 주요 특징 중 하나로 응용프로그램으로 하여금 데이터를 간결하고 효과적인 다차원 희소 배열로 저장 가능하게 하며 Caché 에서는 이러한 배열을 **글로벌(Global)** 이라고 한다.

이 장에서는 다음의 주요 구성에 대하여 설명한다:

- 글로벌의 의미 및 데이터 연산
- 글로벌의 논리적, 물리적 구조
- 어플리케이션에서의 글로벌 저장 및 검색

## 1. 특징

글로벌은 다차원 배열로 데이터를 저장하는 편리한 방법을 제공한다. 다음은 Settings 라는 글로벌의 키 값 “Color”에 데이터 “Red”를 저장하는 예제인데 특별한 사전 정의없이 Caché ObjectScript 명령 SET 을 사용하여 해당 값을 설정하면 된다:

```
SET ^Settings("Color")="Red"  
WRITE !,^Settings("Color")
```

이와 동일한 방법으로 보다 복잡한 다차원 구조도 정의할 수 있다:

```
^Settings("Auto1","Properties","Color") = "Red"  
^Settings("Auto1","Properties","Model") = "SUV"  
^Settings("Auto2","Owner") = "Mo"  
^Settings("Auto2","Properties","Color") = "Green"
```

다음은 글로벌의 일반적 특징들이다:

- **사용의 용이성:** 글로벌은 Caché 클래스 및 루틴에서 변수로 정의없이 바로 사용할 수 있다.
- **다차원:** 위의 예제와 같이 복수의 첨자를 사용하여 다차원 배열의 글로벌을 정의할 수 있다. 첨자로 사용되는 데이터는 정수, 수치, 문자열등 어떠한 타입도 가능하며, 또한 그 값들이 연속적일 필요도 없다.
- **희소(Sparse) 구조:** 글로벌 노드의 저장 공간을 미리 정의할 필요없이 데이터가 생성될 때 데이터의 크기만큼의 저장 공간만이 확보된다. 따라서 매우 공간 집약적이다.
- **효율성:** 삽입, 갱신, 삭제, 검색등과 같은 글로벌 데이터의 연산은 최대 성능 및 병행 작업을 위해 최적화되어 있다. 또한, 대용량 데이터 제어 및 데이터 정렬등의 효율적인 작업을 위한 명령어 및 추가적인 데이터 구조의 글로벌들을 제공한다.
- **신뢰성:** 논리적 및 물리적 저널링을 비롯하여 저장된 글로벌 데이터의 신뢰성을 보장하는 메커니즘을 제공한다.

- **분산:** 글로벌 데이터의 물리적 위치를 제어하는 방법을 제공하여 복수의 데이터베이스에 글로벌을 분산, 저장할 수 있다. 데이터베이스의 위치는 로컬 뿐만 아니라 네트워크 및 어플리케이션 서버 시스템을 통한 공유도 가능하다. 추가로, 샌드위ング 기술을 사용하여 글로벌 데이터를 원격 시스템에 자동 복사할 수도 있다.
- **병행성:** 글로벌 데이터는 다수의 프로세스에 의해 동시에 액세스 가능한데 데이터의 논리적 무결성(integrity)를 위한 강력한 Lock 기능을 제공한다. Caché 오브젝트 및 SQL의 경우, Lock을 통한 병행성이 자동 제어된다.
- **트랜잭션:** 트랜잭션의 시작, 커밋, 롤백 작업을 위한 명령어를 제공한다. Lock과 함께 트랜잭션 로직을 정의함으로써 글로벌 데이터에 대한 ACID 트랜잭션을 구현할 수 있다. 여기서, ACID은 트랜잭션 속성으로 Atomicity(원자성), Consistency(일관성), Isolation(고립성), Durability(영속성)을 의미한다.

다음의 예를 통하여, 글로벌 데이터 사용의 용이성과 빠른 액세스 특성을 간단히 보여주고자 한다. 우선, 첫번째 예는 10,000 개의 노드 배열을 가진 글로벌을 정의하고 저장한 후, 실행된 시간을 측정하는 예이다:

```
SET start = $ZH // get current time
KILL ^Test.Global
FOR i = 1:1:10000 {
    SET ^Test.Global(i) = i
}

SET elap = $ZH - start // get elapsed time
WRITE "Time (seconds): ",elap
```

다음은 저장된 글로벌 데이터의 각 노드를 반복적으로 읽고, 읽은 노드수와 시간을 측정하는 예이다:

```
SET start = $ZH // get current time
SET total = 0
SET count = 0

// get key and value for first node
SET i = $Order(^Test.Global(""),1,data)

WHILE (i != "") {
    SET count = count + 1
    SET total = total + data

    // get key and value for next node
    SET i = $Order(^Test.Global(i),1,data)
}

SET elap = $ZH - start // get elapsed time
```

```
WRITE "Nodes:      ",count,!
WRITE "Total:      ",total,!
WRITE "Time (seconds): ",elap,!
```

## 2. 글로벌의 구조

여기에서는 글로벌의 논리적 및 물리적 구조를 설명하고, 글로벌 데이터 액세스에 대해 알아본다.

### 1) 글로벌의 로직 구조

글로벌은 물리적으로 Caché 데이터베이스(CACHE.DAT)에 저장된 다차원 배열의 이름으로서, 어플리케이션은 원하는 글로벌 데이터를 타깃 네임스페이스를 통해 매핑된 데이터베이스에서 액세스 한다.

이 단원에서는 다음과 같은 글로벌 구조 속성에 대해 살펴본다:

- 글로벌 명명 규약
- 첨자 명명 규약 및 제한
- 글로벌 데이터
- 글로벌 첨자
- 글로벌의 조합

#### 글로벌 명명 규약

글로벌의 이름은 그 목적과 용도에 기반하여 정의한다. 글로벌에는 두 종류의 글로벌과 “프로세스 전용 글로벌”이라는 프로세스 기반 변수들의 집합이 있다:

- **글로벌:** 표준 글로벌이라 할 수 있는 것으로, 대부분의 글로벌 데이터를 가리키며 현재의 네임스페이스를 통해 액세스되는 데이터베이스에 존재하는 persistent 다차원 배열을 말한다.
- **확장 글로벌 참조:** 현재의 네임스페이스가 아닌 네임스페이스에서 액세스 가능한 글로벌을 의미한다.
- **프로세스 전용 글로벌:** 해당 글로벌을 생성한 프로세스에서만 액세스 가능한 배열 변수를 말하며 프로세스 종료와 함께 삭제된다.

다음은 글로벌 명명 규약이다:

- 글로벌명은 반드시 “^” 문자로 시작한다. 이 문자로 글로벌과 로컬 변수를 구분한다. 즉, 이 문자가 없으면 로컬변수로 간주한다.

- “^” 다음의 첫 문자는 다음과 같은 문자이어야 한다:
  - 알파벳 또는 “%” 문자 – 표준 글로벌에만 해당. 글로벌명이 “%”로 시작하면 (“%Z”, “%z”는 제외), 해당 글로벌은 Caché 시스템에서 사용하는 시스템 글로벌로서, %SYS 또는 %CACHELIB 데이터베이스에 저장된다.
  - “|” 또는 “[“ 문자는 확장 글로벌 참조 또는 프로세스 전용 글로벌에 사용된다. 자세한 사용예는 아래의 단원에서 설명한다.
- 그 외 글로벌명에 사용되는 문자로는 알파벳, 숫자, 마침표(.) 등이 될 수 있다. “%” 문자는 글로벌명의 첫 문자로만 사용될 수 있고, 마침표(.)는 글로벌명의 마지막 문자로 사용할 수 없다.
- 글로벌명은 처음의 “^” 문자를 제외하고 최대 31 자까지 정의할 수 있다. 그 이상으로 설정할 수는 있지만, Caché 는 처음 31 자 까지만 유효하게 취급한다.
- 글로벌명은 대소문자를 구분한다.

한편, %Z 와 %z를 제외한 %로 시작하는 모든 글로벌과 Z 와 z로 시작하는 글로벌을 제외하고 %SYS 네임스페이스에 속한 모든 글로벌은 Caché 시스템에서 사용하도록 예약되어 있다.

### 글로벌명 예 및 사용 용도

- ^globalname: 표준 글로벌
- ^|"environment"!globalname: 환경 변수를 포함한 확장 글로벌 참조
- ^||globalname: 프로세스 전용 글로벌
- ^|^"!globalname: 프로세스 전용 글로벌
- ^[namespace]globalname: 네임스페이스를 포함한 확장 글로벌 참조
- ^[directory,system]globalname: 네임스페이스 내부 정보를 포함한 확장 글로벌 참조
- ^["^"]globalname: 프로세스 전용 글로벌
- ^["^","]globalname: 프로세스 전용 글로벌

다음은 다양한 글로벌명의 샘플들이다:

```

SET ^a="The quick "
SET ^A="brown fox "
SET ^A7="jumped over "
SET ^A.7="the lazy "
SET ^A1B2C3="dog's back."
WRITE ^a,^A,^A7,!^A.7,^A1B2C3
KILL ^a,^A,^A7,^A.7,^A1B2C3 // 정의한 모든 글로벌 삭제
  
```

## 첨자 명명 규약 및 제한

글로벌은 노드 레벨을 식별하는 다양한 첨자들을 사용할 수 있는데, 다음과 같은 명명 규약을 따른다:

- null 문자열("")을 제외한 모든 숫자 또는 인용 문자열을 사용할 수 있으며 공백, 특수 문자, 유니코드 문자를 비롯한 모든 타입의 문자를 사용할 수 있다. 첨자는 대소문자를 구분한다.
- 산술 및 연결 연산, 앞뒤 0 제거등 표준 수식 표현식은 숫자 첨자 이름에 대해 수행될 수 있다. 문자열 연결은 문자열 첨자 이름에 대해서도 실행 가능하다.
- 첨자 이름은 로컬 또는 글로벌 변수로 설정될 수 있는데, 이 경우 해당 변수는 반드시 미리 정의되어야 한다.

다음은 이런 조건하에 명명된 유효한 첨자들의 샘플이다:

```
SET ^a(1)="So I've "
WRITE ^a(001.00) ; 유효숫자 앞뒤의 0 들은 제거
SET ^a("2")="got " ; 수치값은 숫자나 문자열이나 동일하게 취급
WRITE ^a(2)
SET ^a(1+2)="that "
WRITE ^a(3) ; 첨자 위치에서 산술 연산 수행
SET ^a(1_2)="going "
WRITE ^a(12) ; 첨자 위치에서 연결 연산자 실행
SET ^a(0)="for "
WRITE ^a(00000) ; 유효 0 을 제외하고 여분의 0 은 제거
SET ^b="word",^a(^b)="me, "
WRITE ^a("word") ; 글로벌 변수로 첨자 정의
WRITE !
; line break for readability
SET ^a("short"_"_"word")="which " ; 첨자 위치에서 문자열 연결
WRITE ^a("short word")
SET ^a("!@#$%^&*")="is " ; 구두 문자들로 첨자값 정의
WRITE ^a("!@#"_"$%^&*")
SET ^a(" ")="nice." ; 공백(space)로 첨자값 정의
WRITE ^a(" ")
SET b=$CHAR(960),^a(b)="Carl Spackler" ; 유니코드 문자를 갖는 로컬 변수로 첨자 정의
WRITE ^a(b)
```

## 첨자의 최대 길이 결정

로컬 및 글로벌 변수를 명명하는데 고려하여야 할 최대 문자수 계산 요소들:

- 글로벌명의 길이
- 사용된 첨자 개수
- 첨자가 숫자인지 문자열인지 여부
- 문자열 첨자의 경우, 사용된 문자 집합 인코딩

첨자 크기를 계산하는데에는 다음의 규칙을 따른다:

1. 글로벌 이름: 한 개의 문자당 1 을 더한다.
2. 숫자 첨자: 각 숫자, 부호, 소수점당 1 을 더한다.
3. 문자열 첨자: 한 개의 문자당 3 을 더한다.

문자열 첨자의 실제 길이는 해당 스트링을 인코드하는데 사용된 문자 집합에 따라 달라진다. 첨자값이 ASCII 문자만으로 이루어진 경우, 각 문자당 1 을 더하지만, 멀티바이트 문자(예, 유니코드)를 포함할 때에는, 각 문자당 3 을 더하게 된다. 따라서, 안정적으로, 최대값 3 을 책정하는 것이다.

4. 첨자: 각 첨자당 1 을 더한다.

이렇게 계산된 전체 합계가 511 개 이상이면, 해당 첨자는 로컬 또는 글로벌의 참조로 사용할 수 없다.

**참고:** 그 외에 다음과 같은 요인도 첨자의 길이를 정하는 데 영향을 미친다:

- 단일 첨자로 사용 가능한 수의 최대치는 309 자릿수이다. 그 이상이면 <MAXNUMBER> 오류가 생성된다.
- 로컬 변수의 문자열 첨자로서 사용 가능한 최대 문자 개수는 251 이다. 그 이상이면 <SUBSCRIPT> 오류가 생성된다. 글로벌 변수의 첨자에는 이러한 제한이 적용되지 않는다.

## 글로벌 데이터

단일 글로벌이 포함하고 있는 데이터는 첨자로 식별되는 하나 이상의 노드에 저장된다. 각 노드는 최대 32,767 개의 문자를 가질 수 있다. 일반적으로, 노드는 다음과 같은 구조를 갖는다:

- 문자열 또는 수치 데이터: Caché 유니코드 버전에서는, 문자열 데이터는 네이티브 유니코드 문자를 갖는다.
- 특정 문자로 구분된 여러 필드들이 연결된 단일 문자열:

```
^Data(10) = "Smith^John^Boston"
```

Caché ObjectScript 의 \$PIECE 함수를 사용하여 각 필드값들을 분리, 추출할 수 있다.

- Caché \$LIST 구조에 포함된 복수의 필드들: \$LIST 는 복수의 필드들을 시스템이 정의한 구조로 구성한다. 즉, 필드값들을 구분하기 위해 사용자 정의 구별자를 제공할 필요가 없다.
- null 문자열: 글로벌 첨자 자체가 데이터로 사용되는 경우, 실제 노드에는 데이터를 갖지 않을 수도 있다.
- 비트스트링: 글로벌이 비트맵 인덱스의 일부분인 경우, 노드에 저장된 데이터는 비트스트링이다. 비트스트링은 1 과 0 만을 데이터로 갖는 압축된 문자열로서, \$BIT 함수를 사용하여 비트스트링을 생성할 수 있다.

- 대용량 데이터의 일부분: 오브젝트와 SQL 엔진은 스트림 데이터(BLOB)를 단일 글로벌의 연속적인 노드에 32K 사이즈의 데이터로 분리하여 저장한다. 그렇지만, 사용자는 스트림 인터페이스를 사용하여 스트림 데이터를 액세스하기 때문에, 스트림이 이와 같은 형태로 분리 저장되어 있는지는 알지 못한다. 즉, 하나의 연속적인 데이터로서만 인식한다.

### 글로벌 첨자

글로벌의 노드는 첨자를 가질 수도 있고, 그렇지 않을 수도 있다. 글로벌 노드의 첨자는 설정하는 첨자 개수에 제한을 받는 것이 아니라, 위에서 언급한대로 전체 글로벌 참조 길이에 제한을 받는다. 첨자값은 연속적인 값일 필요는 없다.

다음은 유효한 글로벌 참조의 예이다:

```
WRITE ^Data
WRITE ^Data(1)
WRITE ^Data(1,2,3)
WRITE ^Data("Customer","453-543",56,"Balance")
```

### 글로벌의 조합

글로벌은 노드의 첨자를 정의된 정렬 순서로 저장된다.

일반적으로 어플리케이션은 첨자로 사용되는 값을 정해진 정렬 타입에 따라 전환하여 노드의 정렬 순서대로 저장한다. 예를 들어, SQL 엔진의 경우, 문자열 값에 대한 인덱스를 생성할 때, 모든 문자열 값을 대문자로 변환하고 공백 문자를 문자열 앞에 추가함으로써, 인덱스가 대소문자를 구분하지 않고 (숫자 값이라도 문자열로 변환하여) 텍스트로 정렬되도록 한다.

## 2) 글로벌의 물리적 구조

글로벌 데이터는 최적화된 구조로 사용하여 물리적인 파일(CACHE.DAT)에 저장된다. 이러한 데이터 구조를 관리하는 코드는 플랫폼에 상관없이 최적화되어 있다. 이러한 최적화를 통하여 글로벌 작업이 높은 처리율(단위 시간당 작업 수량), 높은 작업 병행성(동시 사용자의 수), 캐시(cache) 메모리의 효과적인 사용등을 보장하며, 서비스 운영중 지속적인 성능 관련 튜닝 작업(잦은 재생성, 재인덱스, 압축 등)을 수행할 필요가 없다.

글로벌을 저장하는데 사용되는 물리적 구조는 완벽하게 캡슐화되어 있어, 응용프로그램에서 데이터 구조에 대해 전혀 신경쓸 필요가 없다.

글로벌은 디스크상의 일련의 데이터 블록안에 저장되는데, 각 블록의 크기는(대부분 8KB) 데이터베이스 파일이 생성될 때 결정된다. 데이터의 효과적인 액세스를 위해서, Caché 는 B-tree+ 형의 구조를 갖는다. 또한 디스크 I/O 비용을 줄이기 위하여 자주 사용되는 데이터 블록을

메모리에 캐시하여 사용하는 공유 버퍼 영역을 할당한다.

데이터 저장을 위한 B-tree+ 구조는 다른 데이터베이스 제품들도 많이 사용하는 기술이지만, Caché 는 다음과 같이 Caché 만의 고유한 장점들을 갖고 있다:

- 저장 메커니즘이 안전하고 사용하기 쉬운 인터페이스를 통하여 제공된다.
- 첨자와 데이터는 압축되어 디스크 및 메모리 캐시 공간을 절약한다.
- 저장 엔진은 트랜잭션 처리를 위해 최적화 되어 있어 삽입, 갱신, 삭제 작업이 모두 매우 신속하게 이루어 진다. 또한 관계형 시스템과는 달리, 데이터의 증가로 인해 저하되는 성능을 회복하기 위한 별도의 인덱스나 데이터 재생성이 필요없이, 향상 최고의 성능을 유지한다.
- 동시 사용을 극대화하기 위한 저장 엔진의 최적화가 이루어져 있다.
- 데이터는 효과적인 검색을 위해 자동적으로 일정한 범위내에 연속적 그룹을 형성한다.

### 글로벌 저장 방법

글로벌은 데이터 블록에 순차적으로 저장되며, 첨자와 데이터가 동일 블록내에 함께 저장된다. 그렇지만 긴 문자열과 같은 대용량의 데이터의 경우, 두 이상의 블록에 분산 저장되며, 분리된 블록에 대한 포인터는 노드 첨자와 함께 저장된다.

예를 들어, 다음과 같은 글로벌의 경우,

```
^Data(1999) = 100
^Data(1999,1) = "January"
^Data(1999,2) = "February"
^Data(2000) = 300
^Data(2000,1) = "January"
^Data(2000,2) = "February"
```

단일 데이터 블록내에 다음과 유사한 구조로 저장된다:

```
Data(1999):100|1:January|2:February|2000:300|1:January|2:February|...
```

### 3) 글로벌 참조하기

글로벌은 일반적으로 단일 Caché 데이터베이스에 위치하지만 여러 데이터베이스에 분산 저장될 수도 있다. 데이터베이스는 현재 로컬 시스템에 있을 수도 있고 또는 Caché 네트워킹을 통해 액세스 가능한 원격 시스템에 있을 수도 있다. **데이터셋(dataset)** 은 하나의 Caché 데이터베이스(CACHE.DAT)를 갖는 시스템과 디렉터리를 의미한다.

**네임스페이스**는 데이터 셋과 글로벌 매핑의 논리적 정의이다.

단순한 글로벌 참조는 현재 작업중인 네임스페이스에 적용되며, 해당 네임스페이스는 이 참조 데이터를 읽기 위해 네임스페이스에 정의되어 있는 로컬 또는 원격 시스템상의 데이터베이스에 물리적으로 접근한다. 참조하는 글로벌별로 각각 다른 위치 또는 데이터셋에서 데이터를 액세스할 수 있다.

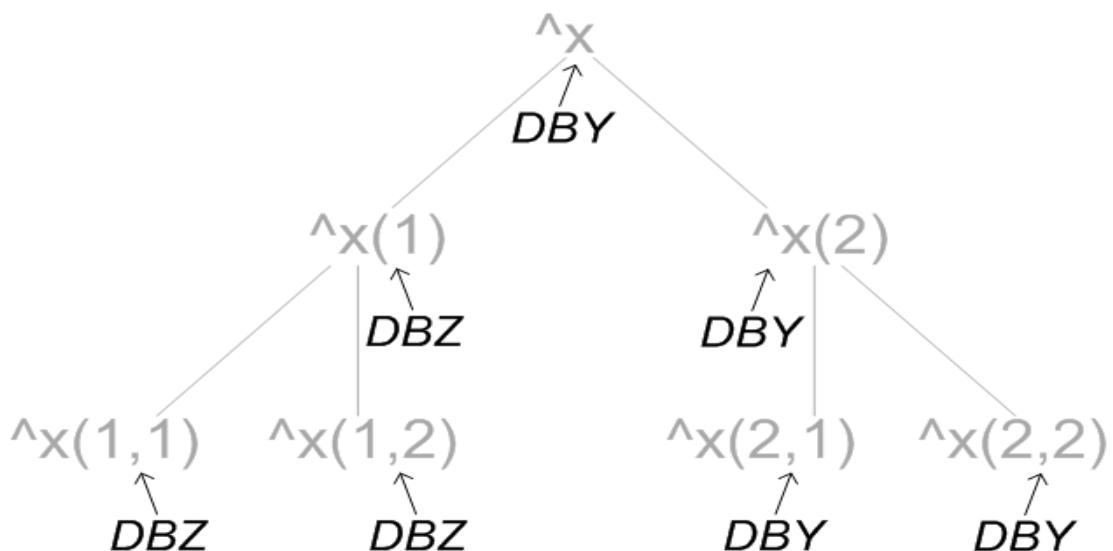
예를 들어, 다음과 같이 글로벌 ORDER 를 참조하는 경우, 글로벌 ORDER 는 현재 네임스페이스에서 액세스 가능한 글로벌이다.

```
^ORDER
```

### 글로벌 매팅

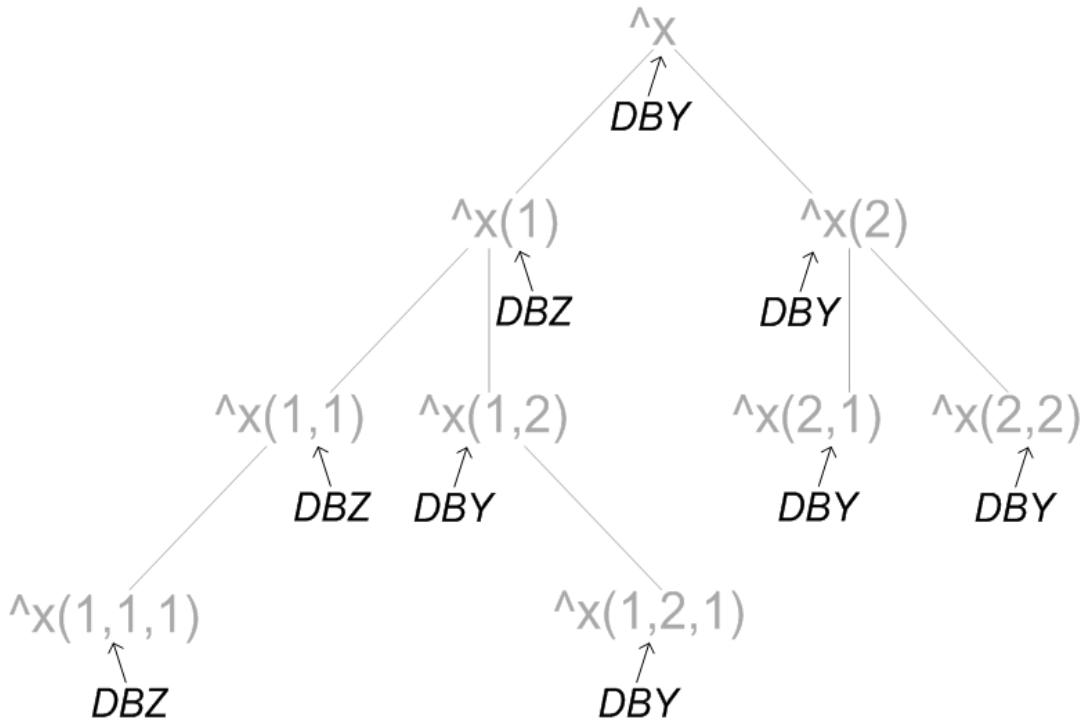
글로벌과 루틴은 하나의 네임스페이스에서 동일 시스템 또는 다른 시스템상의 네임스페이스로 매팅할 수 있다. 즉, 글로벌 매팅을 통해 데이터의 물리적 위치에 상관없이 원하는 데이터를 참조할 수 있으며, 이 매팅 기능이 네임스페이스의 주요 특징이라 할 수 있다. 글로벌 전체 또는 일부분만을 매팅시킬 수 있는데, 글로벌(또는 첨자)의 일부분만을 매팅시키는 것을 첨자 레벨 매팅(Subscript-Level Mapping, SLM)이라 한다. 글로벌 첨자 매팅으로 인하여, 데이터를 쉽게 디스크 전체에 골고루 분산시켜 저장할 수 있다.

글로벌 매팅은 계층적으로 적용된다. 예를 들어, NSX 네임스페이스의 디풀트 데이터베이스가 DBX 라 하자. 이 때,  $^x$  글로벌은 DBY 데이터베이스에,  $^x(1)$  은 DBZ 데이터베이스에 매팅되어 있다면,  $^x(1)$  을 제외한  $^x$  글로벌의 모든 첨자 노드는 DBY 로 매팅되는 반면에, 계층적으로  $^x(1)$  이하의 노드들은 DBZ 로 매팅된다. 다음은 위의 설명에 대한 계층 구조이다:



이 그림에서 글로벌과 그 계층은 회색으로, 매팅된 데이터베이스는 검은색으로 표시되었다.

다음은 보다 복잡한 매핑 구조로서, 글로벌  $\wedge x(1,2)$  노드를 DBY 데이터베이스로 되돌리는 매핑 정의에 대한 그림이다:



이 도표도 앞의 도표와 마찬가지로 글로벌과 그 계층은 회색으로, 대응된 데이터베이스는 검은 색으로 표시되었다.

### 확장 글로벌 참조

글로벌 매핑 정의 없이도 현재의 네임스페이스 이외의 네임스페이스에 위치한 글로벌을 액세스할 수 있는데, 이를 확장 글로벌 참조 또는 간단히 확장 참조라 한다.

확장 참조에는 다음의 두 가지 형태가 있다:

- 명시적 네임스페이스 참조: 글로벌 참조시 확장 형식에 참조하는 글로벌이 위치한 네임스페이스의 이름을 명시한다.
- 암묵적 네임스페이스 참조: 글로벌 참조시 참조하는 확장 형식에 글로벌이 위치한 네임스페이스명을 명시하는 대신, 데이터베이스(CACHE.DAT)가 존재하는 디렉토리를 명시하고, 선택적으로 디렉토리가 존재하는 시스템명을 명시한다. 이 경우, 어떠한 글로벌 매핑도 적용되지 않는데, 이는 직접적인 물리적 데이터설(디렉터리와 시스템)이 글로벌 참조의 일부로 주어졌기 때문이다.

‘명시적 네임스페이스 참조’의 경우, 대상 네임스페이스에서 정의된 글로벌 매핑에 따라 또

다른 데이터베이스에 있는 글로벌들도 참조 가능하다.

‘명시적 네임스페이스 참조’의 사용을 보다 선호하는데, 이는 어플리케이션 코드의 변경없이 필요에 따라 관련 네임스페이스들의 논리적 매핑을 자유롭게 재정의할 수 있기 때문이다.

확장 참조에는 다음의 두 가지 형식이 사용된다:

- **괄호 문법**: 확장 참조를 각 괄호로([ ]) 묶는다.
- **환경 문법**: 확장 참조를 수직 바로(||) 묶는다.

### 괄호 문법

괄호 문법을 사용, 확장 글로벌 참조가 명시적, 또는 암묵적 네임스페이스로 주어진다.

명시적 네임스페이스 참조:

```
^[nspace]glob
```

암묵(시)적 네임스페이스 참조:

```
^[dir,sys]glob
```

명시적 네임스페이스 참조에서, **nspace**는 현재 글로벌 **glob** 를 포함하고 있는 네임스페이스를 말한다. 묵시적 네임스페이스 참조에서의 **dir** 은 참조하고자하는 글로벌을 포함하고 있는 데이터베이스(CACHE.DAT) 파일이 위치한 디렉토리이며, **sys** 는 해당 디렉토리를 갖고 있는 시스템의 이름이다.

암묵적 네임스페이스는,

- 주어진 시스템상의 대상 디렉터리를 참조하거나
- 시스템명이 주어지지 않은 경우, 로컬 시스템에서 대상 디렉터리를 참조할 수 있다. 로컬 시스템의 대상 디렉토리만이 주어질 때는, 디렉토리명 앞에 “^^” 문자를 추가한다.

원격 시스템에 대한 암묵적 네임스페이스 참조:

```
["dir","sys"]
```

로컬 시스템에서의 암묵적 네임스페이스 참조:

```
["^^dir"]
```

예를 들어, 다음은 시스템 “SALES” 에서의 디렉토리 “C:WBUSINESS” 에 위치하는 Caché 데이터베이스 CACHE.DAT 파일에 포함되어 있는 글로벌 ORDER 를 변수 x 에 설정하는 예이다:

```
SET x = ^["C:WBUSINESS","SALES"]ORDER
```

반면에, 다음은 로컬 시스템의 디렉토리 C:WBUSINESS에서 글로벌 ORDER를 액세스하는 예이다:

```
SET x = ^["^^C:WBUSINESS"]ORDER
```

다음은 네임스페이스 “MARKETING”에 속한 글로벌 ORDER에 대한 명시적 네임스페이스 참조의 예이다:

```
SET x = ^["MARKETING"]ORDER
```

## 환경 문법

환경 문법은 다음과 같이 정의된다.

```
^|"env"|global
```

여기에서 **env**는 다음 네 가지 형식중의 하나의 값을 가진다:

- **null ("")** : 로컬 시스템상의 현재 네임스페이스
- **“namespace”** : 참조하는 글로벌이 존재하는 네임스페이스. 네임스페이스명은 대소문자 구별하지 않는다.
- **“^^dir”** : 참조하는 글로벌을 포함하는 Caché 데이터베이스 파일 CACHE.DAT이 존재하는 로컬 시스템상의 디렉터리.
- **“^system^dir”** : 참조하는 글로벌을 포함하는 Caché 데이터베이스 파일 CACHE.DAT이 존재하는 원격 시스템상의 디렉터리.

다음은 로컬 시스템상의 현재 네임스페이스에 존재하는 글로벌 ORDER에 액세스하는 예로서,

```
SET x = ^|""|ORDER
```

다음의 일반적인 글로벌 참조와 동일하다:

```
SET x = ^ORDER
```

액세스하고자 하는 글로벌 ORDER가 네임스페이스 “MARKETING”을 통해 액세스 가능하다면 아래와 같이 액세스할 수도 있고,

```
SET x = ^|"MARKETING"|ORDER
```

글로벌 ORDER를 실제적으로 포함하고 있는 Caché 데이터베이스 CACHE.DAT 파일이 로컬 디렉토리 “C:WBUSINESS”에 존재한다면 다음과 같이 직접 디렉토리를 지정할 수도 있다:

```
SET x = ^|^"^^C:WBUSINESS"|ORDER
```

디렉토리 C:WBUSINESS 가 로컬이 아닌 원격 시스템 “SALES”에 존재한다면 다음과 같이 시스템명을 설정한다:

```
SET x = ^|^SALES^C:WBUSINESS|ORDER
```

### 3. 글로벌의 다차원 저장 방법

이 장에서는 다차원 저장(글로벌 변수)를 제어하는데 필요한 다음과 같은 여러 기능들에 대해 설명한다:

- 글로벌에 데이터 저장하기
- 글로벌 노드 삭제하기
- 글로벌 노드 존재 여부 검사하기
- 글로벌 노드 값 검색하기
- 글로벌에서 데이터 찾아가기
- 글로벌에서 데이터 복사하기
- 글로벌 내보내기
- 글로벌 가져오기
- 글로벌에서 공유 카운터 관리하기
- 임시 글로벌 사용하기
- 글로벌 데이터 정렬하기
- 글로벌 간접 명령 사용하기
- 트랜잭션 관리
- 병행 처리 관리
- 최신 글로벌 참조 체크하기

#### 1) 글로벌에서 데이터 저장

글로벌 노드에서 데이터를 저장하는 것은 다른 변수와 마찬가지로 간단하다. 차이는 글로벌에 대한 작업은 자동적으로 데이터베이스에 저장된다는 점이다.

##### 글로벌 생성하기

글로벌을 생성하는데에는 특별히 추가적인 사항은 필요하지 않으며 변수에 값을 설정하듯 주어진 글로벌(또는 글로벌 첨자 노드)명에 Caché ObjectScript 명령어 **SET** 을 사용하여 데이터를 설정하면 된다:

```
SET ^Color = "Red"
```

이 때, 글로벌 ^Color 가 미리 정의되어 있을 필요는 없다. 즉, 이미 존재하여 값을 갖고 있다면 그 값은 “Red”로 대체될 것이고, 존재하지 않는 글로벌인 경우, “Red”를 설정할 때 자동적으로 글로벌 ^Color 이 생성된다.

### 글로벌 문자 노드에서의 데이터 저장

글로벌 문자 노드에 값을 저장하기 위해서는, 다른 변수 배열처럼 단순히 대상 글로벌 노드에 데이터를 설정하면 된다. 명시된 노드가 존재하지 않는 노드라면 데이터 설정과 함께 생성될 것이고, 이미 존재하는 노드라면 노드에 저장되어 있던 값은 설정하는 새로운 값으로 바꾸어 진다.

글로벌 노드의 값을 설정하는 것은 독립된 단위의 작업으로 병행성을 보장받기 위해 Lock 을 사용할 필요가 없다. 다음은 글로벌 노드에 값을 설정하는 몇몇 유효한 표현식들이다:

```
SET ^Data = 2
SET ^Data("Color")="Red"
SET ^Data(1,1)=100 // ^Data(1)은 정의되지 않은 상태에서 바로 두번 째 문자 레벨인
// ^Data(1,1) 예(은 100 의) 값을(으로) 설정.
// (첫 째 층 문자는 정해지지 않았음)

SET ^Data(^Data)=10 // 글로벌 변수 ^Data 의 값으로 은 문자 노드를 정의(의 이름임)
// 만약, ^Data="a" 라면, 이 연산을 수행한 후, ^Data 의 값은
//      ^Data="a"
//      ^Data("a")=10
// 이다.
SET ^Data(a,b)=50 // 로컬 변수 a 와 b의 값으로(은) 문자 노드를 정의(이름임)
SET ^Data(a+10)=50 // a+10 이 먼저 수행된 후, 그 결과값으로 노드값이 결정된다.
```

### 글로벌 노드에 구조화된 데이터 저장

각각의 글로벌 노드는 최대 32K 의 단일 문자열을 값으로 가질 수 있다. 이 때, 저장되는 데이터는 다음과 같은 형식을 취할 수 있다:

- 최대 32K(32K-1) 크기의 단일값 문자열.
- 의미가 있는 여러 데이터를 특정 구별자로 구분하여 단일 문자열을 구성한다.

여러 필드들을 구별자로 구분하여 단일 문자로 저장하기 위해 각 필드값들을 연결하는데는 Caché ObjectScript 연결 연산자 “\_” 를 사용한다. 다음은 구별자 # 를 사용하여 여러 필드를 연결, 단일 문자열로 구성하는 예이다:

```
SET ^Data(id)=field(1)_"#"_field(2)_"#"_field(3)
```

이와 같이, 구별자를 사용하여 구성된 문자열에서 해당 구별자로 구분된 각 필드값들을 추출하려면 Caché ObjectScript 함수 **\$PIECE**를 사용한다:

```
SET data = $GET(^Data(id))
FOR i=1:1:3 {
    SET field(i) = $PIECE(data,"#",i)
}
QUIT
```

- **\$LIST** 관련 시스템 함수들을 사용하여 복수의 필드들을 단일 문자열로 저장할 수도 있다.  
**\$LIST** 함수는 필드들을 구분하기 위한 구별자를 미리 정의할 필요가 없이 자동적으로 각 필드들을 구분하여 저장한다(이 문자열 구조는 Caché 오브젝트 및 SQL 이 사용하는 기본 구조이다).

다음은 **\$LISTBUILD** 함수를 사용하여 글로벌 노드의 문자열을 정의하는 예이다:

```
SET ^Data(id)=$LISTBUILD(field(1),field(2),field(3))
```

저장된 노드에서 필드별 데이터들을 추출하고자 할 때에는 **\$LIST** 또는 **\$LISTGET** 함수를 사용한다:

```
SET data = $GET(^Data(id))
FOR i = 1:1:3 {
    SET field(i)=$LIST(data,i)
}
QUIT
```

- (스트림이나 BLOB 와 같이) 대용량의 데이터를 일부분씩 분리하여 저장할 수도 있다.

위에서 설명한대로, 하나의 글로벌 노드는 최대 32K의 단일 문자열까지 저장할 수 있기 때문에, 이보다 더 큰 데이터는 다음과 같이, 연속적으로 일련의 노드를 정의하여 저장하게 된다:

```
SET ^Data("Stream1",1) = "스트림의 첫 부분"
SET ^Data("Stream1",2) = "스트림의 둘째 부분"
SET ^Data("Stream1",3) = "스트림의 셋째 부분"
```

Caché 오브젝트 모델에서의 스트림 데이터타입(%GlobalCharacterStream)도 실제로는 위와 같이 일련의 글로벌 노드를 구성하여 스트림 데이터를 저장한다.

- 비트 문자열로 저장하기

비트값(True(1) 또는 False(0)) 값을 데이터로 비트 문자열을 구성, 데이터를 저장할 수 있는데, **\$BIT** 관련 시스템 함수들을 사용하여 액세스한다. 비트 함수들에 대한 자세한

정보는 온라인 문서 Caché ObjectScript References 를 참조하기 바란다.

- 빈 노드로 저장하기

의미있는 실제 데이터가 노드 첨자값 자체인 경우, 일반적으로 해당 글로벌 노드 값은 null("")로 설정한다. 예를 들어, 첨자값이 특정 ID 값과 연결되는 인덱스의 경우, 일반적으로 다음과 같은 모양을 갖는다:

```
SET ^Data("APPLE",1) = ""
SET ^Data("ORANGE",2) = ""
SET ^Data("BANANA",3) = ""
```

## 2) 글로벌 노드 삭제하기

단일 글로벌 노드, 복수의 하위 노드 또는 특정 글로벌 자체를 제거하는데는, Caché ObjectScript 명령 **KILL** 또는 **ZKILL** 을 사용한다.

**KILL** 명령은 주어진 글로벌 노드를 비롯하여 이 노드의 하위에 위치한 모든 노드(데이터뿐 아니라 배열에 대응하는 개체)를 삭제한다. 즉, 주어진 첨자로 시작하는 모든 노드가 삭제된다:

```
KILL ^Data
```

위의 예은 글로벌 ^Data 및 그 하위노드 전체를 삭제한다. 삭제 후, 액세스를 시도하게 되면 <UNDEFINED> 오류가 생성된다.

다음 명령은,

```
KILL ^Data(100)
```

글로벌 ^Data 의 노드 100 만을 삭제하게 되는데, 이 때 ^Data(100,1), ^Data(100,2), ^Data(100,1,2,3) 과 같은 하위 노드들도 함께 삭제된다.

반면에, **ZKILL** 명령은 주어진 글로벌 또는 글로벌 첨자 노드만을 삭제할 뿐, 하위 노드는 삭제하지 않는다.

## 3) 글로벌 노드 존재 여부 테스트

**\$DATA** 함수를 사용하여 특정 글로벌(또는 그 하위 노드)이 데이터를 가지고 있는지 여부를 테스트할 수 있다.

다음 테이블은 \$DATA 함수의 리턴값 및 그 의미이다:

상태	의미
0	주어진 글로벌 변수가 정의되어 있지 않다.
1	주어진 글로벌 변수가 존재하고 데이터를 갖고 있지만 하위 노드는 존재하지 않는다. null 문자열("")도 존재하는 데이터임을 유의하여야 한다.
10	주어진 글로벌 변수 자체는 데이터를 갖고 있지 않지만, 해당 글로벌의 하위 노드들이 존재한다. 예를 들어, ^x(1)은 존재하지만, ^x는 정의되어 있지 않았을 때 \$DATA는 10을 리턴한다.
11	주어진 글로벌 변수 자체 뿐만 아니라 그 하위 노드도 존재하는 경우이다.

#### 4) 글로벌 노드 값 검색

특정 글로벌 노드의 값을 읽기 위하여 다음과 같은 명령어들을 사용한다. 아래의 예에서는, 각각의 단일 명령으로 글로벌 노드 ^Data("Color")의 값을 데이터베이스로부터 메모리로 로드하여 각각의 목적에 따라 사용한다:

```
SET color = ^Data("Color") ; 로컬 변수 color에 값을 설정(할당함)
WRITE ^Data("Color") ; 현재의 디바이스에 바로 출력(명령어에 인수로 사용함)
SET x=$LENGTH(^Data("Color")) ; 함수의 파라미터로 사용(함)
```

#### \$GET 함수

\$GET 함수를 사용 글로벌 노드 값을 읽어올 수도 있다:

```
SET mydata = $GET(^Data("Color"))
```

이 함수를 사용하면, 주어진 글로벌 노드가 정의되어 있지 않은 경우에도 null 문자("")를 반환한다. 또한, 함수의 둘째 인수를 사용하여, 정의되지 않은 노드가 주어졌을 때, 리턴되는 디폴트 값을 정의할 수도 있다. 다른 명령어들을 사용하여 정의되지 않은 글로벌 노드를 액세스하는 경우, <UNDEFINED> 오류가 발생한다.

#### WRITE, ZWRITE, ZZDUMP 명령어

여러 가지 디스플레이 명령어를 사용하여, 글로벌 노드의 값을 출력할 수 있는데, **WRITE** 명령은 주어진 글로벌 노드값을 문자열로 반환하며, **ZWRITE** 명령은 주어진 글로벌 노드 및 하위 노드

전체의 노드명과 노드값을 출력한다. 그리고 ZZDUMP 는 주어진 글로벌 노드 값을 16 진수값의 덤프 형식으로 출력한다.

## 5) 글로벌 노드 데이터 찾아가기

글로벌의 특정 노드에 저장된 데이터를 찾아가기 위해서는 주어진 글로벌 노드에서부터 그 이하의 하위 노드까지의 경로를 배열 첨자 레벨을 따라 반복적으로 거쳐가야 하는데 이러한 작업을 수행하기 위해 제공되는 다양한 방법들에 대해서 알아본다.

### \$ORDER 함수

Caché ObjectScript에서는 \$ORDER 함수를 사용하여, 글로벌의 각 노드를 순차적으로 거쳐갈 수 있도록 하였다. \$ORDER 함수는 현재 레벨의 입력받아 동일 레벨의 다음 첨자값을 반환하며 마지막 첨자값을 입력하면 null("")값을 리턴한다.

다음의 글로벌을 예로 보자:

```
SET ^Data(1) = ""
SET ^Data(1,1) = ""
SET ^Data(1,2) = ""
SET ^Data(2) = ""
SET ^Data(2,1) = ""
SET ^Data(2,2) = ""
SET ^Data(5,1,2) = ""
```

첫번째 첨자 레벨의 첫 첨자값을 찾기 위해서는, 다음과 같이 글로벌 ^Data 의 첫번째 첨자 레벨에 null("")값을 입력한다:

```
SET key = $ORDER(^Data(""))
```

이 때, null 값은 해당 첨자 레벨의 첫번째 값을 얻기 위하여 가장 작은 값의 의미로 주어지는 초기값이다. 만약, 첫 첨자값을 추측할 수 있다면, 그 보다 작은 초기값을 설정함으로써 첫번째 값을 얻을 수 있다. 예를 들어, 첨자값이 양의 정수들로 정의되었다면, 초기값을 0 으로 설정함으로써 첫번째 첨자값을 정확히 얻을 수 있다. 위의 명령을 실행하면, key 값으로 1 이 설정된다. 그러면, 다시 변수 key 를 입력값으로 줌으로써, 반복적으로 다음 첨자값을 얻을 수 있다:

```
SET key = $ORDER(^Data(key))
```

만약, key 값이 1 인 경우, 이 명령문은 2 를 리턴하고, 변수 key 는 새로운 값 2 로 설정된다 (\$ORDER 함수는 동일 레벨의 첨자만을 거치므로, ^Data(1) 다음의 동일 레벨은 ^Data(2) 가 된다). 명령문을 다시 실행하면 key 값은 5 가 된다. 노드 ^Data(5) 자체는 정의되어 있지 않지만, 어쨌든 하위 레벨을 위해 첫번째 레벨의 첨자값 5 는 존재하므로 따라서 5 가 반환된다는 점에 유의하기 바란다. 다시 한번 더 실행하면, key 는 null 값을 리턴받는데, 이는 입력 첨자값 5 가

첫번째 첨자 레벨의 마지막 첨자값이기 때문이다.

이와 같은 방법으로, 첨자를 추가로 사용하여 다른 첨자 레벨도 탐색할 수 있는데, 다음의 예처럼, `^Data(1,"")` 은 첫번째 레벨 첨자값 1 의 두번째 레벨 첨자들을 탐색한다. 이 경우의 리턴 첨자값은 각각 1, 2 가 된다:

```
SET key = $ORDER(^Data(1,""))
```

### \$ORDER 함수의 반복 실행

다음은 글로벌을 정의하고 첫째 레벨 첨자 노드 전체를 함수 `$ORDER` 를 사용하여 반복적으로 탐색하는 일반적인 코드 샘플이다:

```
// 글로벌 ^Data 초기화
KILL ^Data

// 샘플 데이터 생성
FOR i = 1:1:100 {
    SET ^Data(i) = ##class(%PopulateUtils).Name()
}

// 첫 노드 찾기
SET key = $Order(^Data(""))

// 마지막 노드(null 값 리턴)까지 반복
WHILE (key != "") {
    // 리턴된 첨자값 출력
    WRITE "#", key, " ", ^Data(key),!

    // 리턴된 첨자값의 다음 노드 찾기
    SET key = $Order(^Data(key))
}
```

### \$ORDER 함수의 추가 인수

`$ORDER` 함수는 선택적으로 두번째 및 세번째 인수를 설정할 수 있는데, 두번째 인수는 글로벌 노드 검색 방향을 결정하는 방향 플래그이다. 즉, 1 을 주면, 현재의 첨자값을 기준으로 그 다음 첨자값을 찾고, -1 을 주면, 현재 첨자값의 이전 값을 찾는다(즉, 역방향으로 진행). 디폴트값은 1 이다.

세번째 인수는 로컬 변수를 설정하는 것인데 만약, 함수 `$ORDER` 가 리턴하는 첨자값의 노드가 데이터를 갖고 있는 경우, 이 데이터를 주어진 로컬 변수에 설정한다. 글로벌을 검색하여 노드값과 첨자값을 동시에 액세스하고자 할 때 유용하다

## \$QUERY 함수

\$ORDER 함수가 동일 레벨 첨자들만을 검색하기 때문에 주어진 글로벌 노드 이하의 모든 레벨의 모든 첨자값들을 검색하기 위해서는 여러종의 반복 작업(nested loops)을 수행하여야 한다. 이는 추가적인 코드와 프로그램 로직의 복잡성을 야기하기 때문에, 이렇듯 주어진 글로벌 전체 노드를 검색하고자 하는 경우를 위해, Caché ObjectScript 는 \$QUERY 라는 함수를 추가로 제공한다.

\$QUERY 함수는 주어진 글로벌 참조를 인수로 받아 동일 레벨 첨자 여부에 상관없이 해당 글로벌의 다음 노드의 글로벌 참조 문자열을 리턴한다. \$ORDER 함수와 마찬가지로 마지막 글로벌 노드가 주어졌을 때에는, null("") 값을 리턴한다. \$QUERY 함수는 글로벌 노드의 참조 문자열(글로벌명+첨자값)를 리턴하기 때문에 이 문자열값을 액세스하기 위해서는 리턴된 노드 참조값에 Caché ObjectScript 간접명령 연산자(@)를 적용하여야 한다.

다음과 같은 구조의 글로벌에 대해서,

```
SET ^Data(1) = "One"
SET ^Data(1,1) = "OneOne"
SET ^Data(1,2) = "OneTwo"
SET ^Data(2) = "Two"
SET ^Data(2,1) = "TwoOne"
SET ^Data(2,2) = "TwoTwo"
SET ^Data(5,1,2) = "FiveOneTwo"
```

다음의 명령은

```
SET node = $QUERY(^Data(""))
```

로컬 변수 **node**에 해당 글로벌의 첫번째 노드 참조 문자열인 “^Data(1)”를 리턴받아 설정한다. 이 후, 아래의 명령을 반복적으로 실행하면, ^Data(1,1), ^Data(1,2), ^Data(2), … 등이 차례로 리턴된다:

```
SET node = $QUERY(@node)
```

다음은 \$ORDER 를 사용하여 위의 ^Data 글로벌의 범위 전체를 검색하여, 각 노드의 첨자 문자열을 출력하는 예이다:

```
// 첫번째 노드 찾기
SET node = $Query(^Data(""))
WHILE (node != "") {
    WRITE node,!

    // 다음 노드 찾기
    SET node = $Query(@node)
}
```

## 6) 글로벌 데이터 복사하기

주어진 글로벌 노드 전체(하위노드 포함)의 데이터 및 배열 구조를 복사하는데는 Caché ObjectScript 의 **MERGE** 명령어를 사용한다.

다음의 예는 **MERGE** 명령을 사용하여OldData 글로벌의 하위 노드를 포함한 모든 데이터를 newData 글로벌로 복사하는 것이다. 이 때, 배열 구조 또한 동일하게 복사된다.

```
MERGE ^NewData = ^OldData
```

데이터를 복사받는 대상 글로벌이 특정 글로벌의 하위 노드인 경우에도 이 노드에 복사되는 글로벌의 최상위 노드가 설정되고 이하 모든 하위노드들이 동일 배열 구조로 복사된다. 다음 코드는,

```
MERGE ^NewData(1,2) = ^OldData(5,6,7)
```

`^OldData(5,6,7)` 및 그 이하 노드의 모든 데이터를 `^NewData(1,2)`로 복사하는 예이다.

## 7) 글로벌 내보내기

다음의 절차를 통해 글로벌 데이터를 외부 파일로 내보내기할 수 있다:

- 시스템 관리 포탈의 메인 페이지의 데이터 관리 영역에서 “글로벌”을 클릭하여 글로벌 페이지 **[홈] > [글로벌]**로 이동한다.
- 글로벌 페이지에서 “내보내기” 메뉴를 선택하여 글로벌 내보내기 페이지 **[홈] > [글로벌] > [글로벌 내보내기]**를 오픈한다.
- 내보내기 하려는 글로벌이 속해있는 네임스페이스나 데이터베이스를 선택한다:
  - 네임스페이스 또는 데이터베이스의 라디오 버튼을 선택한다.
  - 목록에서 원하는 네임스페이스 또는 데이터베이스를 클릭한다.
- 글로벌을 내보내기할 대상 파일을 경로를 포함하여 입력한다. 또는 검색 버튼을 클릭하여 파일 탐색기에서 파일을 선택할 수도 있다.
- 생성될 파일의 문자 집합을 드롭다운 목록에서 선택한다.
- 글로벌 목록에서 내보내기할 글로벌들을 선택한 후, ‘내보내기’ 버튼을 클릭한다.

## 8) 글로벌 가져오기

다음의 절차를 통해 외부 파일로부터 글로벌 데이터를 가져올 수 있다:

- 시스템 관리 포탈의 메인 페이지의 데이터 관리 영역에서 “글로벌”을 클릭하여 글로벌 페이지 **[홈] > [글로벌]**로 이동한다.

2. 글로벌 페이지에서 “가져오기” 메뉴를 선택하여 가져오기 페이지 [홈] > [글로벌] > [글로벌 가져오기] 를 오픈한다.
3. 글로벌을 가져와 저장할 네임스페이스 또는 데이터베이스를 선택한다:
  - 네임스페이스 또는 데이터베이스의 라디오 버튼을 선택한다.
  - 목록에서 원하는 네임스페이스 또는 데이터베이스를 클릭한다.
4. 글로벌 데이터를 포함하고 있는 가져오기할 파일을 경로명과 함께 입력한다. 또는 검색 버튼을 클릭하여 파일 탐색기로부터 파일을 선택할 수도 있다.
5. 가져올 파일의 문자 집합을 드롭다운 목록에서 선택한다.
6. 파일에 포함된 글로벌 목록을 보기 위해 “열기” 버튼을 클릭한다.
7. 출력된 테이블에서 가져오기 원하는 글로벌(또는 ‘모두 선택’ 버튼)을 선택한다.
8. 가져오기 작업을 백그라운드에 실행하려면 백그라운드 실행 선택 상자를 체크한다.
9. ‘가져오기’ 버튼을 클릭한다.

## 9) 글로벌 공유 카운터 관리

대용량 트랜잭션 처리 응용프로그램에서의 주된 병행 처리 병목 현상 요인은 유일 식별자 값의 생성이라 할 수 있는데, 예를 들어, 주문 처리 어플리케이션의 경우, 새로운 청구서별로 고유의 식별 번호를 필요로 할 것이다. 이러한 데이터의 전형적인 관리 방법은 일종의 카운터 테이블을 사용하는 것이다. 이 때, 새로운 주문 청구서를 발생하는 모든 프로세스는 이 카운터 값에 대한 Lock 을 얻기 위해 기다려야 하고, Lock 을 얻은 후 해당 값을 증가시킨 후 다시 Lock을 해제해야 한다. 이러한 과정에서 이 단일 리소스에 대한 과도한 리소스 경쟁을 초래할 수 있다.

이러한 문제를 해결하기 위해, Caché ObjectScript 는 **\$INCREMENT** 함수를 제공한다. **\$INCREMENT** 함수는 글로벌 노드의 값을 원자적(atomic)으로 1 씩 증가시킨다(글로벌 노드가 정의되어 있지 않은 경우에도, 초기값 null 로부터 1 을 설정함). **\$INCREMENT** 의 이러한 원자적(atomic) 특성은 Lock 을 적용할 필요가 없으며, 따라서 다른 프로세스의 영향을 받지 않고 새로운 증가 값의 설정을 보장해 준다는 것이다.

다음은 **\$INCREMENT** 함수의 사용 예이다. 우선 카운터를 적요할 글로벌 노드를 정하고, 해당 글로벌의 새로운 카운터 값이 필요할 때마다, 이 글로벌을 인수로하여 **\$INCREMENT** 함수를 호출한다:

```
SET counter = $INCREMENT(^MyCounter)
```

Caché 오브젝트 와 SQL 에서 사용하는 기본 저장 방식은 고유의 오브젝트(행) 식별자 값을 부여하기 위하여 **\$INCREMENT** 함수를 사용한다.

## 10) 임시 글로벌 사용하기

특정 작업에 있어서는 데이터를 데이터베이스에 저장할 필요는 없지만 글로벌의 장점들을 필요로 할 수도 있다. 예를 들어, 디스크에 저장할 필요는 없지만 글로벌의 강력한 정렬 연산을 이용하기 위하여 데이터를 글로벌로 정의하는 경우를 생각할 수 있다. 이런 경우를 위하여, Caché 에서는 임시 글로벌을 제공하고 있다. 임시 글로벌은 다음과 같은 특징을 가진다:

- 임시 글로벌은 항상 CACHETEMP 데이터베이스에 저장되는데, 이 데이터베이스는 로컬 데이터베이스로 정의된다. CACHETEMP 데이터베이스로 매핑되는 모든 글로벌은 임시 글로벌로 취급된다.
- 임시 글로벌의 변경 사항은 디스크에 기록되지 않고 메모리 공유 버퍼 영역에 머물게 된다. 만약 메모리 버퍼 영역에 충분한 공간이 없으면 대용량 임시 글로벌 순으로 디스크에 임시로 저장될 수도 있다.
- 효율성 극대화를 위해, 임시 글로벌에 대한 변경은 저널링되지 않는다.
- 임시 글로벌은 Caché 시스템이 다시 시작할 때마다 자동 삭제된다.

디폴트로 “CacheTemp”로 시작하는 모든 글로벌명은 임시 글로벌로 규정한다. Caché 자체에서 사용할 수 있는 임시 글로벌과의 충돌을 피하기 위해 사용자 정의 임시 글로벌명은 CacheTempUser로 시작하도록 명명하기를 권장한다.

Caché SQL에서는 임시 글로벌을 복잡한 쿼리의 최적화를 위한 보조 메모리 공간으로 사용하며 또한, 특정 쿼리 작업(정렬, 분류, 집합체 계산 등)을 위한 임시 인덱스로서 사용하기도 한다.

## 11) 글로벌 데이터 정렬

글로벌에 저장되는 데이터는 문자 값에 따라 자동적으로 정렬된다. 예를 들어, 다음의 코드에서와 같이 일련의 글로벌들을 정의하면, 정의된 순서에 관계없이, 문자값의 알파벳 순서(“Athens”, “Boston”, “Cambridge”, “London”, “New York”)로 글로벌 노드가 자동적으로 정렬되어 있는 것을 볼 수 있다:

```
// 기존 데이터 삭제
KILL ^Data

// 글로벌 노드 정의
SET ^Data("Cambridge") = ""
SET ^Data("New York") = ""
SET ^Data("Boston") = ""
SET ^Data("London") = ""
SET ^Data("Athens") = ""

// 정의된 글로벌 노드 참조값 저장되어 있는 순서대로 출력
SET key = $Order(^Data(""))
WHILE (key != "") {
```

```

    WRITE key,!  

    SET key = $Order(^Data(key)) // 다음 첨자 읽기  

}

```

응용프로그램에서 특정 값들의 순서나 인덱스 참조등과 같은 정렬 작업이 필요한 경우, 글로벌의 자동 정렬 기능을 활용하면 손쉽게 작업을 수행할 수 있다.

### 글로벌 노드의 조합

글로벌 노드가 정렬되는 순서는 두 단계로 제어되는데 하나는 글로벌 자체이고 다른 하나는 해당 글로벌을 사용하는 응용프로그램에 의해서이다.

응용프로그램 단계에서는 첨자로 사용되는 값에 대한 데이터 변환을 수행할 때 글로벌 노드의 조합을 제어할 수 있다(Caché SQL과 오브젝트에서는 사용자가 지정한 조합 함수를 사용한다). 예를 들어, 다음의 코드와 같이 대소문자를 구분하지 않는 알파벳 순서대로 정렬되는 첨자값들의 목록을 만들기 위하여, 해당 첨자값을 대문자로 변환하는 함수(\$ZCONVERT)를 사용하여 그 값을 저장하고 해당 글로벌 노드 값은 변환하지 않은 원래값으로 정의하여 보자:

```

// 기준 데이터 삭제
KILL ^Data

// 글로벌 노드 데이터 정의
FOR name = "Cobra","jackal","zebra","AARDVark" {
    // 첨자값을 대문자로 변환하여 저장
    SET ^Data($ZCONVERT(name,"U")) = name
}

// 저장되어 있는 순서대로 반복해서 읽기
SET key = $Order(^Data(""))
WHILE (key != "") {
    WRITE ^Data(key),!      // 변환되지 않은 글로벌 노드 데이터 출력
    WRITE key,!             // 변환된 글로벌 노드 첨자값 출력
    SET key = $Order(^Data(key)) // 다음 글로벌 노드
}

```

### 수치 및 문자열 첨자

숫자 값은 문자열 값보다 앞에 위치하게 된다. 즉, 첨자값 1은 문자 “a”보다 앞에 놓이게 된다. 따라서, 동일 레벨의 첨자값으로 숫자와 문자열을 함께 사용하는 경우, 이 점을 고려하여야 한다. 인덱스를 정의하는 경우(즉, 데이터를 값을 기준으로 정렬), 정렬 대상 데이터들은 숫자(월급 등)나 문자열(우편번호 등)인 경우가 대부분이다.

수치적으로 정렬되는 노드에 있어서, 주어진 첨자값을 수치값으로 확인, 변환할 때 사용하는 대표적인 방식은 해당 첨자값에 강제적으로 단항 “+” 연산자를 적용하는 것이다. 예를 들어,

아래의 예와 같이, **id** 값을 **age** 을 기반으로 정렬하는 인덱스를 생성하는 경우, 강제적으로 **age** 값에 강제적으로 “+” 연산자를 적용함으로써, 항상 숫자 값이 적용되도록 보장할 수 있다:

```
SET ^Data(+age,id) = ""
```

반면에 수치적 의미의 값(“0022”, “0342”, “1584”)을 문자열로 정렬하려면, 공백 문자(“ ”)를 주어진 문자값 앞에 강제적으로 첨부함으로써 문자열화할 수 있다. 예를 들어, 아래의 예와 같이 **id** 값을 문자열 **zipcode** 기반으로 정렬하는 인덱스를 생성하는 경우, **zipcode** 값이 항상 문자열로 취급되도록 “ ” 문자를 **zipcode** 앞에 추가한다:

```
SET ^Data(" " _zipcode,id) = ""
```

이렇게 함으로서, “0022” 와 같이 0으로 시작하는 값들을 문자열로 취급할 수 있게된다.

### \$SORTBEGIN, \$SORTEND 함수

일반적으로 Caché 에서의 데이터 정렬은 개발자가 고려하지 않아도 될 사항인데, 이는 SQL 를 사용하건 글로벌을 직접 액세스하건 상관없이, 정렬은 자동적으로 이루어지기 때문이다.

그렇지만 데이터 정렬을 보다 효과적으로 수행하여야 할 경우가 발생할 수도 있는데, 특히 1) 무작위로 대용량의 글로벌 노드를 정의할 때 라든지, 2) 특정 작업 – 인덱스 재생성, 인덱스되지 않은 임시 데이터의 정렬 등등 – 의 결과가 너무 많아 Caché 버퍼 공유 영역의 대부분을 차지하여 시스템 성능에 중대한 영향을 미치는 경우등을 들 수 있다.

이러한 경우를 효과적으로 처리하기 위하여, Caché ObjectScript 에서는 **\$SORTBEGIN** 및 **\$SORTEND** 함수를 제공하고 있다. **\$SORTBEGIN** 함수는 주어진 글로벌 전체(또는 그 일부) 데이터를 메모리(또는 임시 디스크 저장소)내의 특정 버퍼 영역에 로드하여 해당 영역의 배타적 사용을 설정한다. 정렬 작업이 완료되면, **\$SORTEND** 함수는 정렬된 순서에 따라 데이터를 실제 글로벌 저장소(데이터베이스)에 저장한다. 이는 보다 적은 횟수의 디스크 입출력 작업으로 이루어지기 때문에 전체적으로 매우 효율적인 작업을 구현할 수 있게 된다.

**\$SORTBEGIN** 및 **\$SORTEND** 함수은 아래의 예와 같이 매우 간단하게 사용할 수 있다:

```
// 기존 데이터 삭제
KILL ^Data

// ^Data 글로벌 정렬 모드 시작
SET ret = $SortBegin(^Data)

// ^Data에 랜덤 데이터 적기
FOR i = 1:1:10000 {
    SET ^Data($Random(1000000)) = ""
}
```

```

SET ret = $SortEnd(^Data)

// ^Data 가 현재 설정되어 정렬됨
// 반복하고 디스플레이함(순서대로)
SET key = $Order(^Data(""))
WHILE (key != "") {
    WRITE key,!
    SET key = $Order(^Data(key)) // 다음 문자
}

```

**\$SORTBEGIN** 함수는 특별한 경우의 글로벌 생성을 위해 디자인되었기 때문에, 사용 시 주의할 사항이 있다. 글로벌이 **\$SORTBEGIN** 모드에 있을 때는 글로벌 데이터에 대한 읽기를 시도하지 말아야 하는데, 이는 데이터가 실제 데이터베이스에 저장된 것이 아니기 때문에, 데이터의 최종 값을 보장할 수 없기 때문이다.

Caché SQL 에서는 인덱스하지 않은 필드들의 정렬과 같은 임시 인덱스 글로벌 생성등에 자동적으로 이 함수들을 사용한다.

## 12) 글로벌에 대한 간접 명령 사용

Caché ObjectScript 에서는 간접 명령을 제공함으로써, 실행 시간에 글로벌 참조의 동적 생성을 구현할 수 있다. 이는 프로그램 컴파일 타임에 알 수 없는 글로벌명이나 구조를 실행 시간에 정의하여야 할 때, 유용하게 사용할 수 있다.

간접 명령은 간접 명령 연산자 "@" 를 통해 이루어지며, "@" 를 사용하는 용도에 따라 몇 가지의 타입으로 분류된다.

다음은 글로벌 참조를 데이터로 갖는 문자열에서 해당 글로벌 참조를 추출하기 위해 @ 연산자를 사용하는 예이다:

```

// 기존 데이터 삭제
KILL ^Data

// 글로벌 참조 표현식을 로컬변수에 설정
Set var = "^Data(100)"

// 간접 명령을 사용하여 ^Data(100) 에 노드값 설정
SET @var = "이 자료는 간접적으로 설정됨"

// ^Data(100)에 설정된 값 보기
WRITE "Value: ",^Data(100)

```

또는 다음과 같이, 표현식(변수 또는 실제 값)을 혼합한 문자 간접 명령문을 사용할 수도 있다:

```

// 기존 데이터 삭제
KILL ^Data

// 글로벌명을 변수에 설정
SET glvn = "^Data"

// 간접 명령을 사용, ^Data(1)에서 ^Data(10) 까지 설정
FOR i = 1:1:10 {
    SET @glvn@(i) = "이 데이터는 간접적으로 설정됨"
}

// 설정된 값 보기
SET key = $Order(^Data(""))
WHILE (key != "") {
    WRITE "Value ",key, ":", ^Data(key),!
    SET key = $Order(^Data(key))
}

```

간접 명령은 Caché ObjectScript의 주요 특징 중 하나로서 글로벌 참조에만 적용이 국한되는 것은 아니다. 그렇지만 데이터의 직접 액세스보다 다소 비효율적이기 때문에, 사용에 각별히 주의하여야 한다. 간접 명령에 대한 보다 자세한 정보는 온라인 문서 Using Caché ObjectScript 의 “Operators”장을 참조하기 바란다.

### 13) 트랜잭션 관리

Caché 는 글로벌을 사용하는 트랜잭션 프로세싱의 전과정을 제어하는데 필요한 원천적인 연산들을 제공하고 있다. Caché 오브젝트 와 SQL 에서는 이러한 트랜잭션 제어가 자동적으로 이루어 지는데, 이러한 제어 연산을 글로벌 데이터에 적용하여 직접 트랜잭션 로직을 구현할 수도 있다.

트랜잭션 명령어에는 트랜잭션 시작을 정의하는 TSTART, 현재의 트랜잭션을 완료하는 TCOMMIT, 그리고 현재의 트랜잭션을 취소하고 트랜잭션 시작 이후 변경된 모든 글로벌 데이터를 트랜잭션 이전 값으로 되돌리는 TROLLBACK 등이 있다.

다음은 트랜잭션 시작을 정의하고 ok 값에 따라 트랜잭션을 커밋하거나 롤백하는 예이다:

```

TSTART

SET ^Data(1) = "Apple"
SET ^Data(2) = "Berry"

IF (ok) {
    TCOMMIT
}
ELSE {

```

## TROLLBACK

}

**TSTART** 는 트랜잭션 시작 표시를 Caché 저널 파일에 기록함으로써 트랜잭션의 시작 시점을 설정해 준다. 위의 예에서, 변수 `ok` 가 `true` (1 또는 0 이외의 값) 이면, **TCOMMAND** 은 트랜잭션의 성공적 종료를 기록하고 완료 표시를 저널 파일에 기록된다. `ok` 가 `false` (0) 이면 **TROLLBACK** 은 트랜잭션 시작 이후의 모든 SET 및 KILL 명령으로 변경된 값을 트랜잭션 이전 상태의 값으로 재설정된다. 위의 예에서는, `^Data(1)`, `^Data(2)` 값이 TSTART 이전 값으로 복귀한다.

### 잠금과 트랜잭션

트랜잭션의 고립성(isolation) – 트랜잭션이 완료되기 전에는 트랜잭션에서 변경된 내용을 다른 프로세스에서 액세스 할 수 없는 것 – 을 보장하려면 Lock을 같이 사용하여야 한다. Caché ObjectScript 의 LOCK 명령을 사용하여 글로벌에 대한 잠금 및 해제를 시행한다.

또한 트랜잭션내에서 적용된 잠금은 트랜잭션이 종료(커밋 또는 룰백)될 때까지 해제되지 않는데, 다음의 절차를 통해서 그 이유를 설명하고자 한다:

1. TSTART 를 사용 트랜잭션 시작한다
2. 수정하려는 노드에 Lock 를 적용한다. 보통 쓰기 작업에 잠금(write lock)을 적용한다.
3. 해당 노드를 수정한다.
4. Lock 을 해제한다. 트랜잭션 종이므로, 이 시점에서 Lock이 해제되지는 않는다.
5. TCOMMIT 을 사용, 트랜잭션을 종료한다. #4 에서 적용된 Lock 해제가 이 시점에서 비로소 실질적으로 해제된다.

만약 다른 프로세스에서 이 트랜잭션에서 커밋되지 않은 변경 데이터를 제외한 노드들을 보고자 할 경우에는, 데이터를 읽기 전, 해당 노드에 대하여 Lock(read lock)을 테스트해 볼 수 있다. ‘write lock’ 은 트랜잭션 종료 시까지 해제되지 않기 때문에, ‘read lock’ 을 통해서는 트랜잭션이 종료될 때까지(커밋 또는 룰백) 데이터를 읽을 수가 없다.

대부분의 데이터베이스 관리 시스템들은 트랜잭션의 고립성을 제공하기 위하여 유사한 메커니즘을 사용하고 있다. 하지만 Caché는 추가적으로 개발자들로 하여금 그들만의 유용하고 고유한 로직을 구현할 수 있는 기법을 제공하고 있다. 즉, 트랜잭션을 실행하면서도 동시에 새로운 어플리케이션에 맞는 사용자 정의 데이터베이스 구조를 생성할 수 있다는 것이다. 물론 Caché 오브젝트나 SQL을 사용하여 시스템으로 하여금 트랜잭션을 자동 관리하게 할 수도 있다.

## TSTART 의 다중 적용

Caché는 \$TLEVEL이라는 시스템 변수를 사용하여, 현재 TSTART 명령이 몇 번이나 호출되었는지 추적하여 알려준다. \$TLEVEL 값은 0 부터 시작하여, TSTART 가 호출될 때마다 1씩 증가하며, TCOMMIT 이 실행될 때마다 1씩 감소한다. 마지막 TCOMMIT 의 호출로 \$TLEVEL 값이 다시 0으로 설정되면, 마지막 트랜잭션도 커밋과 함께 종료되는 것이다.

TROLLBACK 명령이 실행되면, 현재의 모든 트랜잭션은 종료되며, 현재 값에 관계없이 \$TLEVEL 값은 0으로 설정하게 된다.

이러한 특성은 응용프로그램으로 하여금 자체적으로 트랜잭션을 갖는 코드(예, 오브젝트 메소드) 전체를 또 다른 트랜잭션 단위로 정의하게 함으로써 보다 큰 단위의 비즈니스 트랜잭션 로직을 구현할 수 있게끔 한다. 예를 들어, persistent 오브젝트를 저장할 때 사용되는 %Save 메소드는 자체적으로 독립된 트랜잭션을 수행하는데, 어플리케이션 레벨에서 이 메소드를 포함한 비즈니스 로직 단위에 TSTART 및 TCOMMIT 명령을 명시적으로 사용한다면, 일련의 개별 오브젝트 저장 작업들을 포함하는 보다 큰 규모의 트랜잭션을 구현할 수 있는 것이다:

```
TSTART
SET sc = object1.%Save()
IF ($$$ISOK(sc)) {
    // 위의 첫 %Save() 가 성공하면, 다음의 두번째 %Save() 가 시작
    SET sc = object2.%Save()
}

IF ($$$ISERR(sc)) {
    // 두 %Save() 중 하나라도 실패하면, 뒤로
    TROLLBACK
}
ELSE {
    // 두 %Save() 모두 성공하면. 커밋
    TCOMMIT
}
```

## 14) 병행 처리 관리

단일 글로벌 노드에 대한 설정 또는 검색 작업은 독립된 단위로 이루어 지는데, 이는 항상 일관된 결과로 실행을 완료한다. 반면에 복수의 노드에 대한 작업 또는 트랜잭션 고립성(transaction isolation)을 제어하기 위해서는, Lock 기능을 사용하여 글로벌 노드에 대한 잠금 및 해제를 적용함으로써 병행 처리를 관리한다.

Lock 은 Caché Lock 관리자에 의해 관리되며, Caché ObjectScript 코드에서는 LOCK 명령을 사용하여, 글로벌 노드에 대한 잠금 및 해제를 직접 제어한다.

## 15) 최근 참조 글로벌 확인

가장 최근에 참조된 글로벌은 Caché ObjectScript 의 시스템 변수 **\$ZREFERENCE** 에 기록된다. **\$ZREFERENCE** 는 첨자 및 확장 글로벌 참조를 포함한 전체 참조 문자열을 포함한다. 그러나 **\$ZREFERENCE** 이 갖고 있는 글로벌 참조는 해당 노드의 액세스에 대한 성공 여부 또는 존재 여부에 상관없이, 단순히 가장 최근에 참조된 글로벌 참조명일 뿐이다.

### 단축 글로벌 참조(Naked Global Reference)

첨자로 구성된 글로벌 참조에 있어서, Caché 는 해당 글로벌명 및 첨자 레벨에 대한 단축 표시자(naked indicator)를 설정한다. 그러면 동일 글로벌명과 첨자레벨을 사용하는 경우에, 후속 참조에 있어서는 글로벌명과 상위 레벨의 첨자를 생략한 단축 글로벌 참조를 사용하여도 후속 노드들을 참조할 수 있다. 이는 동일 글로벌의 동일 첨자 레벨에 대한 반복적 액세스를 효율적으로 실행할 수 있도록 하여준다.

단축 참조에서 하위 첨자 레벨을 설정함으로써 첨자 레벨에 대한 단축 표시자값을 재설정하기 때문에 단축 글로벌 참조를 사용할 때면, 항상 가장 최근의 글로벌 참조로 만들어는 첨자 레벨에서 작업을 수행하는 것이다.

단축 표시자 값은 **\$ZREFERENCE** 시스템 변수에 기록되며, 단축 표시자가 설정되어 있지 않을 때 단축 글로벌 참조를 시도하면, <NAKED> 오류가 발생한다. 네임스페이스를 바꾸면 단축 표시자는 null("")로 초기화되며, **\$ZREFERENCE** 값을 null 문자열("")로 직접 설정하여도 단축 표시자를 초기화할 수 있다.

다음의 예는, 첨자 글로벌 ^Produce("fruit",1)를 첫번째 참조로 설정한 경우이다. Caché는 이 글로벌명과 첨자를 단축 표시자에 저장함으로써, 후속 글로벌 참조에서는 글로벌명 "Produce" 와 상위 첨자 레벨인 "fruit" 을 생략하고 바로 단축 참조 ^(3,1) 만을 사용하여도 하위 첨자 레벨로 진행되며, 이 새로운 첨자 레벨이 후속 단축 글로벌 참조의 새로운 단축 표시자가 되는 것이다.

```
SET ^Produce("fruit",1)="Apples" /* 전체 글로벌 참조 */
SET ^(2)="Oranges"             /* 단축 글로벌 참조 */
SET ^(3)="Pears"               /* 첨자 레벨 2 가정*/
SET ^(3,1)="Bartlett pears"   /* 첨자 레벨 3 으로 이동 */
SET ^(2)="Anjou pears"         /* 첨자 레벨 3 가정 */
WRITE "최신 글로벌 참조 ",$ZREFERENCE,!
ZWRITE ^Produce
KILL ^Produce
```

위의 예에서는 다음과 같은 글로벌 노드가 생성된다: ^Produce("fruit",1), ^Produce("fruit",2),

`^Produce("fruit",3), ^Produce("fruit",3,1), ^Produce("fruit",3,2).`

거의 예외 없이, 모든 글로벌 참조는 단축 표시자를 설정한다. **\$ZREFERENCE** 시스템 변수는 단축 글로벌 참조인 경우에도, 전체 글로벌명 및 첨자를 가진다. 또한 **ZWRITE** 명령도 단축 참조의 사용 여부와 관계없이 글로벌의 전체 이름과 첨자를 모두 출력한다.

단축 글로벌 참조는 조심스럽게 사용되어야 하는데, 다음과 같은 경우를 포함하여 항상 분명한 상황에서 단축 표시자를 설정하는 것은 아니기 때문이다.

- 전체 글로벌 참조는 최초로 단축 표시자를 설정하며, 후속의 전체 글로벌 참조나 단축 글로벌 참조는 글로벌 참조 액세스의 성공 여부에 상관없이 단축 표시자의 값을 변경한다. 예를 들어, 존재하지 않는 글로벌에 대한 **WRITE** 명령은 **<UNDEFINED>** 오류를 리턴하지만 단축 표시자의 값은 액세스하고자 하였던 글로벌 값으로 변경된다.
- 첨자 글로벌을 참조하는 후조건 명령문의 경우에도, 조건문의 계산 결과와는 관계없이, 단축 표시자의 값을 변경한다.
- 첨자 글로벌을 참조하는 선택적인 함수 인수들은 Caché 가 해당 인수들을 계산하느냐에 따라, 단축 표시자를 설정할 수도, 안 할 수도 있다. 예를 들어, **\$GET** 의 두번째 인수는 디폴트 값이 사용되지 않더라도 항상 단축 표시자를 설정한다. Caché 는 인수를 왼쪽에서 오른쪽으로 계산하기 때문에, 마지막 인수가 첫번째 인수로 인해 이미 설정된 단축 표시자를 재 설정할 수도 있다.
- **TROLLBACK** 명령은 단축 표시자에 적용되지 않는다. 즉, 트랜잭션내에서 단축 표시자의 값이 변경되었다 하더라도, 룰백시 트랜잭션 이전의 값으로 다시 설정되지는 않는다.

전체 글로벌 참조가 확장 글로벌 참조를 포함하고 있는 경우, 후속의 단축 글로벌 참조는 동일한 확장 글로벌 참조를 가진다고 추정하기 때문에 확장 참조를 단축 글로벌 참조에 포함시킬 필요가 없다.

# 제 7 장 Caché Server Page에 의한 Web 어플리케이션 구축

## 1. Caché Server Pages(CSP) 소개

- 1) CSP 및 Zen
- 2) 시작하기 전에
- 3) 첫번째 CSP 페이지 생성

## 2. CSP 아키텍쳐

- 1) CSP 구성 요소: 웹 서버, CSP Gateway, CSP 서버
- 2) 웹 서버 URL 구성
- 3) CSP Gateway 구성
- 4) CSP 어플리케이션 옵션

## 1. Caché Server Pages(CSP) 소개

Caché Server Pages(CSP)는 Caché 기반 웹 어플리케이션을 구축할 때 사용되는 구조이자 툴셋이다. CSP 기술을 사용하면 고성능의 확장성이 우수한 웹 어플리케이션을 구축하고 구현할 수 있다. CSP는 일반적으로 Caché 데이터베이스의 데이터를 사용하여 동적으로 웹 페이지를 생성한다. “동적으로”라는 의미는 페이지 요청 시 최근 변경된 데이터 소스가 동일한 페이지에 바로 적용되어 출력될 수 있다는 의미이다.

CSP는 다음과 같은 기능들을 제공한다. (즉,)

- 변경 데이터를 실시간으로 액세스 한다.
- 수천의 동시 사용자를 갖는 웹 기반 시스템도 지원 가능하다.
- Caché 데이터베이스에 저장된 사용자 정보(속성 및 보안 허용 수준)를 토대로 페이지를 특정 사용자에 맞게 정의 구성할 수 있다.
- HTML, XML, 이미지 또는 기타 바이너리 및 텍스트 데이터를 제공한다.
- 고성능 Caché 데이터베이스에 밀접하게 연결되어 있어 동일한 빠른 성능을 제공한다.

CSP는 내장된 Caché 데이터베이스에 신속하게 액세스할 수 있어 데이터베이스 어플리케이션에 매우 적합하며 또한 웹 기반 데이터 어플리케이션에 필수적인 다음과 같은 다양한 기능들을 제공한다.

- 세션 관리
- 페이지 인증
- 웹 페이지에서 대화식으로(interactive) 데이터베이스 작업 수행

CSP는 다음 두 가지 웹 개발 유형을 지원한다.

- 클래스를 사용하여 어플리케이션을 개발할 수 있도록 객체 프레임워크를 제공한다.
- HTML 파일을 사용하여 어플리케이션을 개발할 수 있도록 웹 페이지 안에 객체 및 서버 기반 스크립트를 포함시킬 수 있는 HTML 마크업 언어를 제공한다. 사용자는 한 어플리케이션 안에서 이 두 가지 기술을 결합할 수 있기 때문에 개발 유연성을 극대화할 수 있다.

### 1) CSP 및 Zen

Zen은 풍부한 데이터를 기반으로 하는 웹 어플리케이션을 신속하게 생성할 수 있는 어플리케이션 프레임워크이다. 새로운 웹 기반 어플리케이션을 구축하거나 기존의 CSP 기반 어플리케이션의 업그레이드를 고려 중이라면 Zen 프레임워크를 검토해 보기를 추천한다.

## 2) 시작하기 전에

이 책은 사용자가 웹 서버 및 Caché를 이미 설치한 상태라고 가정한다. 이 장에는 CSP 어플리케이션 생성 준비에 필요한 작업이 설명되어 있다.

### 프로덕션 웹 서버 및 Caché 제공 사설 웹 서버

Caché는 시스템 관리 포털을 실행하기 위한 사설 웹 서버(private web server)를 제공한다. 이 사설 웹 서버를 사용하여 제공된 CSP 샘플이나 CSP 페이지를 실행할 수도 있다. 그러나, 실제 서비스 환경에서의 서비스되는 CSP 어플리케이션에는 사용할 수 없다. 이 때문에, 프로덕션 시스템에서는 지원 가능한 웹 서버(예: Apache 웹 서버, Microsoft의 IIS 웹 서버 또는 Sun 웹 서버)를 반드시 별도로 설치해야 한다.

사설 웹 서버는 최소 규모로 구축된 Apache 웹 서버를 기반으로 한다. 이 서버는 57772 (또는 일반적인 HTTP 서버 포트 80 이 아닌 다른 포트) 와 같은 비 표준 TCP 포트로 구성된다. 사설 웹 서버는 동일한 호스트 상에서 운영되는 다른 웹 서버에 영향을 주지 않는다.

### 웹 서버 및 CSP Gateway 구성

Caché 설치에는 일반 웹 서버와 운영 시스템 기반위에 웹 서버 및 CSP Gateway 구성을 실행하는 스크립트가 포함되어 있다.

대부분의 경우 Caché 지침에 따라 Caché를 설치하고, 지원되는 웹 서버의 일반적 구성을 설치하면 CSP Gateway와 함께 작동하는 시스템이 구축된다.

그러나, 일반적이 아닌 웹 서버 구조를 보유하고 있거나 사용자 정의 환경으로 시스템을 설정하려는 고급 사용자인 경우, Caché 온라인 문서 [CSP Gateway Configuration Guide](#)를 참조하기 바란다. 이 문서에는 웹 서버 및 CSP Gateway를 구성하여 Caché에 연결하는 절차에 대한 세부 사항이 설명되어 있다. 웹 서버 및 CSP Gateway를 설정하여 원격 Caché 서버에 설치된 CSP 어플리케이션에 액세스하는 절차는 [CSP Gateway](#)

Configuration Guide의 “Using Caché Server Pages with a Remote Web Server” 단원을 참조하기 바란다.

### 주요 실행 요구 사항

효율적인 CSP 개발 및 운영을 위해서는, 다음 사항들에 대한 사전 지식이 필요하다:

- Caché 오브젝트 및 Caché ObjectScript
- HTML
- JavaScript
- SQL

HTML 및 JavaScript 기술에 대한 정보는 다음의 사이트에서 얻을 수 있다:

- [HTML v4.0.1 Specification](#)
- [HTML & XHTML: Definitive Guide](#) (발행인: O'Reilly)
- [JavaScript: Definitive Guide](#) (발행인: O'Reilly)

### CSP 샘플

Caché는 CSP 페이지 샘플 세트와 함께 제공된다. 이 샘플을 보려면

1. Caché를 시작한다.
2. 웹 서버가 실행중인지 확인한다.
3. 브라우저를 시작한 후 **CSP Sample Menu**로 이동한다. 포트 번호 57772를 사용하는 사설 웹 서버인 경우,

<http://localhost:57772/csp/samples/menu.csp>

또는 외부 웹 서버의 경우

<http://localhost/csp/samples/menu.csp>

로 이동한다.

4. 일반 또는 잠금 보안 레벨로 Caché를 설치한 경우, 로그인 페이지가 나타날 수도

있다. 이 경우, 로그인한다.

5. Caché는 각각에 대한 간략한 설명과 함께 CSP 페이지 샘플 목록을 표시한다. 관심이 있는 목록을 클릭한다.

## CSP 문서

CSP 온라인 문서는 다음과 같은 정보를 제공한다:

- [Using Caché Server Pages](#)에는 CSP 페이지 생성 방법이 설명되어 있다.
- [Using ZEN](#)에는 미리 구축된 페이지 컴포넌트들을 사용하여 신속하게 Web 어플리케이션을 개발할 수 있도록 CSP 기반 위에서 작동하는 패키지인 ZEN 사용 방법이 설명되어 있다. Zen 문서에는 [Using Zen Components](#), [Developing Zen Applications](#), 및 [Using Zen Reports](#)가 포함되어 있다.
- [CSP HTML Tag Reference](#)는 모든 CSP 태그에 대한 정보를 제공한다.
- [CSP Samples Menu](#)는 다양한 CSP 페이지 샘플들을 보여준다.
- [Caché Server Pages Quick Start Tutorial](#)는 CSP 기본에 대한 빠른 이해를 돋기 위한 자습서이다.
- [CSP Web Applications Tutorial](#)는 심도 깊은 이해를 위한 자습서이다.
- 클래스 참조 정보:
  - [%CSP.Page](#)
  - [%CSP.Session](#)
- *CSP Web Gateway* 문서 – CSP Gateway 구성에 대한 온라인 도움말로서 시스템 관리 포털을 통해 액세스되는 CSP 게이트웨이 관리 페이지에서 이용 가능하다.
- [홈] > [환경구성] 페이지로 이동한 후, CSP 게이트웨이 관리를 선택한 후 도움말을 클릭한다. 기본적으로, 모든 관리 페이지 및 문서는 Caché 웹 서버 포트(예, 57772)를 사용하는 사설 웹 서버로 이동하게 된다. 프로덕션 웹 서버에 대한 CSP 웹 게이트웨이 관리 페이지를 보려면 URL에 localhost:57772 대신에 localhost 또는 localhost:<port\_no>를 사용한다.
  - (예: <http://localhost/csp/bin/Systems/Module.cxw>).

- CSP Gateway Configuration Guide – Caché를 설치할 때 CSP Gateway가 자동으로 설치되며 대부분의 경우 디플트 설정으로 운영 가능하다. 만약 디플트 이외의 CSP Gateway 구성이 필요한 경우, 고급 구성 지침을 사용하여야 한다.

### 3) 첫번째 CSP 페이지 생성

이 장에서는 CSP 페이지 ‘Hello, World’ 를 생성하는 방법을 설명한다.

1. Caché Studio를 시작한다.
2. 메뉴 ‘파일 → 새로 만들기’ 를 선택한 후 ‘새로 만들기’ 창의 ‘CSP 파일’ 탭을 선택한다.
3. ‘CSP 파일’ 탭에서 Cache Server Page 아이콘을 선택한 후 ‘확인’ 버튼을 누른다.
4. Studio 편집창에 다음과 같은 기본 html 태그로 이루어진 CSP 파일이 생성된다:

```
<html>
<head>

<!-- Put your page Title here -->
<title> Cache Server Page </title>

</head>

<body>

<!-- Put your page code here -->
My page body
</body>
</html>
```

5. 텍스트 ‘My page body’ 를 다음과 같이 변경한다:

```
<b>Hello, World!</b>
```

6. 페이지를 Hello.csp 로 저장한다.
7. 메뉴 ‘보기 → 웹 페이지’를 선택하여 웹 브라우저에서 Hello.csp 가 요청되고 Hello,World 가 표시되는 것을 확인한다.

**참고:** 텍스트 에디터나 HTML 에디터를 통해서도 HTML 파일을 생성할 수 있다. 이 때

생성된 웹 페이지는 로컬 디렉터리인 /cachesys/csp/user (cachesys: Caché가 설치된 위치)에 Hello.csp 파일로 저장한다.

CSP 어플리케이션은 다음과 같이 실행된다:

1. 브라우저가 로컬 웹 서버로 Hello.csp에 대한 요청을 전송한다.
2. 웹 서버는 이 요청을 (웹 서버에 연결된) CSP Gateway에 전송하며 이 요청이 Caché CSP 서버로 전달된다.
3. Caché CSP 서버는 Hello.csp 파일을 찾아 CSP 컴파일러로 전송한다.
4. CSP 컴파일러는 Hello.csp 파일의 컨텐츠를 작성하는 **OnPage** 메소드를 포함한 csp.Hello 라는 새로운 클래스를 생성한다 (컴파일러는 실제적으로 OnPage 메소드로부터 호출되는 여러 메소드들을 생성한다).  
웹 브라우저로부터 요청된 .csp 파일이 이미 서버에 생성되어 있는 클래스보다 최신인 경우에만 이 컴파일링 절차가 이루어진다. 이후 요청은 생성된 클래스로 바로 전송된다.
5. CSP 서버가 새로 생성된 **OnPage** 메소드를 실행하고 이 메소드로부터의 출력이 위의 예와 같이 브라우저로 전송된다.

위의 예제는 프로그램적 개발에 있어서 이해를 돋기 위한 매우 간략한 설명일 뿐이며 실제 CSP 컴파일러는 다음과 같은 작업을 수행하도록 전문화된 XML/HTML 처리 엔진이다:

- HTML 페이지에서 서버상의 스크립트 및 표현식 처리
- 특정 HTML 태그에 대응하는 서버 메소드 실행

따라서 다음과 같이 CSP 페이지에 프로그래밍 로직을 삽입하고 실행할 수 있다. 이 때, 이 로직은 서버로 전달되어 실행되고 그 결과를 웹 페이지를 통해 브라우저에 출력한다:

```
<html>
<body>
<b>Hello, World!</b>
<script language="Cache" runat="server">
 // this code is executed on the server
 Write "<ul>", !
 For i = 1:1:10 {
   Write "<li> This is item ", i, !
 }
 Write "</ul>", !
</script>
</body>
</html>
```

위의 코드는, 10 개의 메시지를 연속적으로 웹 페이지에 출력하는 예이다.

## 2. CSP 아키텍쳐

이 장에서는 다음의 내용들을 설명한다:

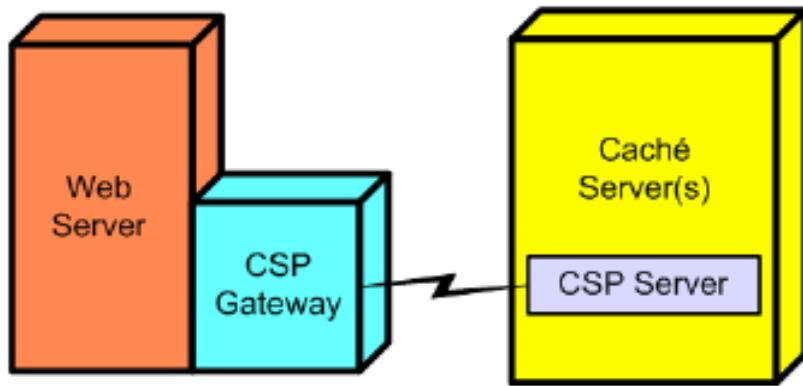
- 주요 “CSP 구성 요소” .
- CSP 사용을 위한 지원되는 “웹 서버 구성”
- Caché 서버와의 통신을 위한 “CSP Gateway 구성”
- “CSP 어플리케이션 옵션”

사용자 웹 서버 환경에서의 CSP 설치 및 구성에 관한 자세한 정보는 온라인 문서 CSP Gateway Configuration Guide 를 참조 바람.

### 1) CSP 구성 요소: 웹 서버, CSP Gateway, CSP 서버

CSP는 다음의 3 가지 주요 컴포넌트로 구성된다 – 웹 서버, CSP Gateway 및 CSP 서버(Caché 서버 상에서 실행) :

## CSP 구성 요소



웹 서버 및 CSP 서버는 단일 또는 복수의 컴퓨터로 구성할 수 있다. 개발 중 세 구성 요소(웹 서버, CSP Gateway 및 CSP 서버) 모두는 단일한 PC 상에 존재할 수도 있으며 큰 규모의 시스템인 경우, 2 단계 또는 3 단계로 구성된 웹 서버 및 CSP 서버를 구축할 수도 있다.

이 책에서는 세 구성 요소(웹 서버, CSP Gateway 및 CSP 서버)를 독립된 각각의 단계로 취급한다. 또한, CSP가 XML 페이지는 물론 다른 텍스트 포맷 및 이미지와 같은 바이너리 포맷도 지원하고 있지만 오직 HTML 페이지만 지원하는 것처럼 설명한다.

### 각 구성요소의 역할

웹 서버는 다음과 같은 역할을 수행한다:

- 브라우저로부터 전달되는 HTTP 요청 수락
- 허가 확인
- 정적(static) 컨텐츠 제공
- CSP Gateway에 CSP 컨텐츠(.csp or .cls로 끝나는 URL) 요청 전송

CSP Gateway는 공유 라이브러리, 즉 .dll 파일 또는 CGI 스크립트으로서 기능은 다음과

같다:

- 요청을 전송할 Caché 서버 결정
- 정확한 Caché 서버로 요청 전송
- 새 연결의 연속적 오픈을 방지하기 위한 Caché 서버와의 기존 연결에 대한 유지관리

CSP 서버는 CSP Gateway에서 전송된 요청을 처리하는 Caché 서버 프로세스로, 기능은 다음과 같다.

- 어플리케이션에 대한 HTTP 요청 수신
- (시스템 관리 포탈에서 설정되고 .cpf 파일에 저장된) 어플리케이션 구성 설정 확인
- (CSP 페이지에서 직접 작성되거나 생성된) 관련 클래스 실행한 후 결과 HTML를 웹 서버 및 브라우저로 리턴하는 CSP Gateway로 전송

## 프로세스 흐름

CSP 요청은 표준 웹 서버(모든 범용 서버 지원) 및 표준 HTTP 프로토콜을 기반으로 처리된다. CSP는 웹 서버와 Caché 간 통신을 관리하며, 어플리케이션 코드를 실행하여 페이지를 생성한다. 요청 및 응답 프로세스는 다음과 같다.

1. HTTP 클라이언트(일반적으로 웹 브라우저)는 HTTP를 사용하는 웹 서버에 페이지를 요청한다.
2. 웹 서버는 이를 CSP 요청으로 인식, 빠른 서버 API를 통해 CSP Gateway로 전송한다.
3. CSP Gateway는 통신할 Caché 서버를 결정하여 해당 시스템의 CSP 서버에 요청을 전송한다.
4. Caché에서 실행 중인 CSP 서버는 요청을 처리하여 결과 페이지를 웹 서버로 리턴한다.
5. 웹 서버는 이 페이지를 출력할 수 있도록 브라우저로 전송한다.

## 정적(Static) 파일

모든 CSP는 Caché 서버에 의해 처리된다. 표준 Web 어플리케이션의 경우, 정적 컨텐츠는 데이터베이스 서버가 아닌 웹 서버를 통해 제공된다. 반면 Caché의 경우 기본적으로 웹 서버는 모든 정적 컨텐츠를 Caché 서버로부터 제공받도록 구성되어 있다. 만약 웹 서버에서 정적 페이지를 제공하는 구성을 사용하는 웹 어플리케이션이 이미 설치된 상태라면 웹 서버로부터 정적 컨텐츠를 제공받는 구성을 선택, 사용할 수도 있다. 그렇지만 정적 페이지를 Caché 서버로부터 제공받는 것이 보다 용이할 뿐만 아니라 단일 웹 서버가 각각 다른 버전의 특정 정적 파일(예: 하이퍼이벤트 브로커 구성 요소)을 필요로하는 서로 다른 버전의 Caché 서버들로 구성되어 있을 때 발생하는 불일치를 해소할 수 있다.

그렇지만 CSP Gateway로 하여금 웹 서버상에 정적 파일들을 캐시하도록 하는 경우, 정적 페이지 요청을 Caché 서버로 하지 않고 웹 서버상에서 바로 처리하므로 속도 향상에 기여할 수도 있다. 보다 상세한 정보는 온라인 문서 Using Caché Server Pages -> CSP Architecture 의 “CSP Application Options” 테이블의 ‘Serve Files’를 참조하기 바란다.

웹 서버가 정적 파일을 제공할 수 있도록 (웹 서버 자체를) 구성하는(한) 경우, 정적 컨텐츠가 모든 웹 서버상에 존재하는지 확인하여야 한다.

## 2) 웹 서버 URL 구성

Caché는 CSP 어플리케이션을 실행하기 위한 디폴트 가상 디렉터리 /csp 와 다수의 Caché 인스턴스를 설치한 경우, 추가적으로 /cacheinstance/csp 라는 가상 디렉터리도 자동적으로 설치 제공한다. 다수의 Caché 인스턴스가 설치되어 있는 경우, /cacheinstance 없이 /csp 를 사용하여 CSP 어플리케이션을 액세스하게 되면 가장 최근에 설치된 Caché 인스턴스로 액세스하게 된다. 만약 모든 CSP 어플리케이션이 <http://localhost/csp> 또는 <http://localhost/cacheinstance/csp> 로 시작되는 어플리케이션 경로로 호출되도록 구성한다면, 웹 서버에 대한 추가적인 구성 변경은 필요하지 않다.

/csp 또는 /cacheinstance/csp 이외의 경로로 시작하는 CSP 어플리케이션을 정의하려면, 웹 서버 구성 파일 입력을 변경해야 한다.

다음 테이블은 변경이 필요한 구성 파일들의 목록이다.

웹 서버	구성 파일
Apache, HP Secure 웹 서버	UNIX: /etc/httpd/conf/httpd.conf Windows: <web-server-install-directory>\conf\httpd.conf OpenVMS: <web-server-install-directory>\CONF\HTTPD.CONF
Sun	config/obj.conf 및 config/magnus.conf
Microsoft IIS	"Microsoft IIS 웹 서버에서 새로운 URL 구성"에 설명된 대로 가상 디렉터리를 정의한다.

다음 테이블은 웹 서버 및 Caché 인스턴스를 사용하여 CSP 페이지에 액세스할 때의 URL 예를 보여주고 있다.

URL	웹 서버	인스턴스
http://localhost/cache20071/csp/samples/menu.csp	공공	cache20071
http://localhost/cache52/csp/samples/menu.csp	공공	cache52
http://localhost/csp/samples/menu.csp	공공	최신 Caché 버전 설치됨
http://localhost:57772/csp/sys/UtilHome.csp	사설 — 시스템 관리 포탈 및 온라인 문서 액세스 시 사용	웹 서버 포트 57772 를 사용하는 Caché

참고: CSP는 대소문자를 구별한다. CSP를 구성할 때 경로 이름을 일관성 있게 규정하여야

한다.

## Microsoft IIS 웹 서버에서 새로운 URL 구성

Microsoft IIS는 일련의 **가상 디렉터리**를 정의하여 구성된다. 각 가상 디렉터리는 이름(URL 중 디렉터리 부분과, 실제 디렉터리(웹 서버에서 정적 파일을 제공하도록 구성하는 경우, .html 또는 .jpg와 같은 정적 파일이 저장될 수 있는 로컬 디렉터리) 및 권한 정의(예: **Read** 또는 **Execute**)로 구성되어 있다.

CSP 컨텐츠에 대한 요청(URL)에는 디렉터리 이름이 포함되어 있다. 이 디렉터리명은 웹 서버상에 정의된 가상 디렉터리 또는 가상 디렉터리의 하위 디렉터리이어야 하며 CSP 컨텐츠가 제공될 수 있도록 최소한 읽기 및 실행(**Read** 및 **Execute**) 권한이 있어야 한다.

웹 서버에서의 정적 파일 제공을 선택하는 경우, 웹 서버가 가상 디렉터리에 대응하는 물리적 디렉터리에서 정적 파일(예: .html 또는 .jpg)을 찾는다. .csp 파일의 경우, 웹 서버 및 Caché 서버 모두 물리적 디렉터리에서 파일을 찾지는 않는다. 즉, .csp 파일은 Caché 서버에 존재하게 된다. 웹 서버 및 Caché 모두 동일한 시스템에서 실행 중인 경우(애플리케이션 개발시 권장 구성), 정적 파일 및 .csp 파일 모두 동일한 위치에서 검색될 수도 있는데 이러한 구성이 바로 Caché를 설치하며 동시에 로컬 웹 서버를 구성하는 방식이다.

Caché는 설치시 IIS 서버가 실행되고 있는지 감지하여 /csp 이름의 가상 디렉터리를 정의하려고 시도한다. 이렇게 /csp 가 정의되면 /csp 의 하위 디렉토리 /csp/samples 및 /csp/user 에 대한 요청이 로컬 Caché 서버로 전송된다.

새로운 CSP 어플리케이션을 추가하는 경우, 이 새로운 어플리케이션에 대한 URL 경로 역시 /csp로 시작하면 IIS 구성 작업을 별도로 할 필요가 없다. 예를 들어, /csp/myapp는 IIS 가상 디렉터리에 정의되어 있는 /csp 를 사용한다. 그렇지만, 새 CSP 어플리케이션이 /csp 로 시작하지 않는 경우, 반드시 해당 어플리케이션 경로에 대응하는 새로운 가상 디렉토리를 IIS 서버에 정의하여야 한다.

예를 들어, URL 경로 /myapp 를 사용하는 CSP 어플리케이션을 정의하려면 다음과 같은 순서로 정의한다.

1. IIS 관리자를 실행한다 (참고: 모든 Windows 버전은 해당 버전마다 고유한 방법으로 IIS 관리자를 실행하지만 일반적으로는 Windows 의 제어판에서 실행한다.)
2. IIS의 로컬 컴퓨터의 ‘기본 웹 사이트’에서 마우스 오른쪽 클릭 후 ‘새로 만들기 -> 가상 디렉토리’를 선택하여 /myapp 라는 가상 디렉토리를 정의한다.  
(이 작업을 수행하는 예를 보려면 Caché 온라인 문서 *CSP Gateway Configuration Guide* 의 “Add Virtual Directories in IIS” 장을 참조).
3. 이 디렉토리에 대한 읽기 및 실행 권한을 설정한다.
4. 웹 서버가 정적 컨텐츠를 제공하길 원하는 경우, 정적 컨텐츠를 저장하려는 물리적 디렉토리를 정의한다.

계속해서 다음 장에 설명된 대로 추가적인 CSP Gateway 및 CSP 서버 구성을 설정한다.

### 3) CSP Gateway 구성

CSP Gateway는 웹 서버에 설치되고 웹 서버에 의해 실행되는 DLL 또는 공유 라이브러리이다. CSP Gateway는 확장자가 .csp 또는 .cls 인 모든 요청을 정의된 Caché 서버로 전송하여 처리하게 한다.

#### CSP Gateway 관리자

CSP Gateway 관리자(CSP Web Gateway 관리 페이지)를 사용하거나 구성 파일 csp.ini 를 직접 편집하여 CSP Gateway를 구성할 수 있다.

CSP Gateway 관리자는 브라우저에서 사용 가능한 소규모의 웹 애플리케이션이다. 시스템 관리 포탈 [홈] > [환경구성] 페이지의 ‘연결성’ 부분에서 CSP 게이트웨이 관리를 선택하면 CSP Web Gateway 관리 페이지에 액세스할 수 있다. 이 경우, 디폴트로 사설 웹 서버를 통해 CSP Gateway 관리자에 액세스한다.

프로덕션 웹 서버를 통해 CSP Gateway 관리자에 액세스하려면 localhost 또는 localhost:<port\_no> 으로 교체하여 URL을 변경한다. 단, *port\_no* 는 프로덕션 웹 서버가 사용하는 포트 번호이다.

CSP Gateway 구성 속성들에 관한 정보는 CSP Web Gateway 관리 페이지의 도움말 링크를 클릭하면 얻을 수 있다. 그 외 보다 자세한 정보는 온라인 문서 *CSP Gateway Configuration Guide* 를 참조하기 바람.

## 서버 액세스 정의

CSP Gateway가 액세스할 수 있는 서버(CSP 어플리케이션을 실행할 수 있는 Caché 또는 Ensemble 서버) 목록을 정의할 수 있다. 각 서버의 기본 정보로는 논리적인 이름, TCP/IP 주소, TCP/IP 포트 번호(기본: 1972) 및 서비스 상태(활성화 또는 안하기) 플래그등이 있다. 그 외에, 세션당 연결 가능한 최소 및 최대 연결 수, 타임아웃 및 이벤트 로그 레벨등도 구성 가능한다.

서버마다 각각의 논리적 이름이 있기 때문에, CSP Gateway는 이 이름을 기준으로 특정 서버에 어플리케이션을 쉽게 연결시킬 수 있으며, 서버를 사용하는 모든 어플리케이션을 재구성할 필요 없이 해당 서버의 특성만을 변경할 수 있다.

처음 설치를 하고 나면, CSP Gateway 에는 Caché의 로컬 인스턴스에 연결되도록 설정된 서버 연결명, **LOCAL** 이 정의되어 있다..

CSP Gateway 에서 액세스 가능한 서버를 추가하려면 위에서 설명된 CSP Gateway 관리자를 연 후 서버 액세스를 클릭한다. 이 절차에 대한 세부사항은 온라인 문서 *CSP Gateway Configuration Guide* 의 “Accessing CSP on Multiple Caché Servers” 장을 참조하기 바란다.

CSP.ini 파일의 디폴트 LOCAL 서버에 대한 예:

```
LOCAL=Enabled  
...  
[LOCAL]  
Ip_Address=127.0.0.1  
TCP_Port=1972  
Minimum_Server_Connections=3
```

## 어플리케이션 액세스 정의

**참고:** CSP는 대소문자를 구별한다. CSP를 구성할 때 일관적으로 경로 이름을 규정한다.

CSP 어플리케이션은 주어진 URL에 의해 실행되어지는 페이지나 클래스 집합이다. 예를 들어, 모든 CSP 샘플 페이지는 /csp/samples 어플리케이션의 구성원이 된다. 정의된 어플리케이션에는 하위 디렉터리가 포함되어 있을 수도 있다(예: /csp/samples/cinema).

CSP Gateway 관리자를 사용하여 CSP 어플리케이션이 특정 Caché 서버에 연결할 때 사용하는 URL을 정의한다. CSP는 특정 URL 디렉터리(그리고 그 하위 디렉터리)에 있는 모든 파일을 동일한 어플리케이션의 일부로 간주한다.

디풀트 구성으로, CSP Gateway는 모든 CSP 요청을 연결 서버명 LOCAL로 전송하는 단일 어플리케이션 경로, /csp 로 정의되어 있다. 따라서 /csp/samples 및 /csp/user 에 대한 요청은 로컬 Caché 서버로 전송된다.

또한 /csp 로 시작하는 URL로 새로운 CSP 어플리케이션을 생성하는 경우, CSP Gateway 구성은 변경할 필요가 없다. 즉, 새로운 어플리케이션(예: /csp/myapp)은 이미 정의되어 있는 /csp 구성을 사용하게 된다. 반면에 해당 URL 경로로 /csp 이외의 다른 것을 원하는 URL 경로와 일치하는 새 CSP 어플리케이션을 CSP Gateway 관리자를 통해 정의하여야 한다.

다음은 URL 경로가 /myapp 로 시작하는 CSP 어플리케이션을 정의하는 순서이다:

1. 시스템 관리 포탈의 **[홈] > [환경구성]** 페이지에서 CSP 게이트웨이 관리를 클릭하여 CSP Gateway 관리자를 오픈한다.
2. **응용프로그램 액세스**를 선택한다.
3. **응용프로그램 추가** 버튼을 클릭한다.
4. **응용프로그램 경로** 필드에 /myapp를 입력한다.
5. 디풀트 서버 목록에서 어플리케이션이 사용할 디풀트 서버명을 선택한다 (디풀트 서버명은 위의 “서버 액세스 정의” (장)에서 정의한다).
6. **환경구성 저장** 버튼을 클릭하여 /myapp 어플리케이션 액세스 구성을 저장한다.

**응용프로그램 액세스** 페이지의 다른 필드들에 대한 자세한 정보는 페이지 오른쪽 상단의 **도움말**을 통해 제공된다.

## CSP Gateway 파라미터

CSP Gateway 는 타임아웃, CGI 환경변수들과 같은 조정 가능한 매개변수들을 제공한다. 제공되는 모든 파라미터들에 대한 자세한 정보는 온라인 문서 CSP Gateway Configuration 의 “Operation and Configuration” 장이나 CSP Web Gateway 관리 페이지의 **도움말**을 통해 얻을 수 있다.

## 4) CSP 어플리케이션 옵션

이 장에서는 CSP 어플리케이션 설정을 통해 Caché가 CSP 어플리케이션 요청을 처리하는 방법이 설명되어 있다. CSP 서버가 HTTP 요청을 받았을 때, 로컬 Caché CSP 구성 설정을 사용하여 요청 처리 방법을 결정한다.

시스템 관리 포탈의 **CSP 응용프로그램** 편집 페이지 상에서 특정 CSP 어플리케이션 처리 방법에 대한 설정을 다음과 같이 변경할 수 있다.

1. [홈] > [보안 관리] > [CSP 응용프로그램] 를 선택하면 현재 정의되어 있는 CSP 어플리케이션의 목록을 볼 수 있다. 타입 칼럼은 어플리케이션을 사용자 어플리케이션(CSP) 또는 시스템 어플리케이션(CSP, System; Caché 시스템이 제공하는 CSP 기반 유ти리티) 인지를 구별한다.
2. 원하는 어플리케이션을 선택한 후 편집을 클릭한다.

다음은 일반탭에서 설정할 수 있는 옵션들이다:

### CSP 어플리케이션 옵션 — 일반 탭

필드	설명
CSP 응용프로그램명 *	어플리케이션의 이름을 입력한다.
행 전체 삭제 - 없어진 옵션	새로운 어플리케이션의 경우, 기존의 어플리케이션을 선택한다. 이 필드는 선택된 어플리케이션의 옵션을 통해 채워지므로 필요에 따라 편집 가능한다.
설명	설명을 입력한다.

사용가능	어플리케이션 이용 가능 여부를 제어한다. 체크했을 경우, 사용자 인증 및 권한 부여에 따라 어플리케이션을 이용할 수 있으며, 체크되지 않았을 경우에는, 해당 어플리케이션을 사용할 수 없다.
응용프로그램 실행에 요구되는 리소스	어플리케이션을 실행할 때 반드시 (역할 권한의 일부분으로서) ‘사용’ 허가되어야 하는 리소스를 설정한다. 자원 및 허가에 관한 정보는 온라인 문서 <i>Caché Security Administration Guide</i> 의 “About Resources” 장을 참조.
허용된 인증 메소드	<p>어플리케이션 실행에 필요한 인증 방법을 규정한다. 어플리케이션이 복수의 인증 방법을 지원하는 경우, 다음과 같이 인증이 이루어진다.</p> <ul style="list-style-type: none"> <li>‘인증 안함’을 포함한 복수의 옵션이 가능한 경우 사용자는 사용자명과 패스워드없이 로그인하는 방법을 선택할 수도 있다.</li> <li>모든 옵션이 사용 가능하고 사용자명과 패스워드를 사용하는 경우, Caché는 주어진 순서에 따라 사용하여야 하는 인증 절차를 수행한다. 이용 가능한 인증 방법은 <b>인증 옵션</b> 페이지([홈] &gt; [보안 관리] &gt; [시스템 보안 설정] &gt; [인증 옵션])에서 선택하여 결정된다. 인증 순서에 대한 정보는 온라인 문서 <i>Security Administration Guide</i> 의 Authentication 장의 Cascading Authentication 부분을 참조.</li> <li>선택된 옵션이 커베로스(Kerberos) 및 Caché 로그인(‘인증 안함’은 포함 안됨)인 경우 사용자명과 패스워드는 반드시 입력해야 한다. 이 경우, Caché는 먼저 커베로스 인증을 수행한 후 Caché 로그인을 시도한다. 하나의 인증이라도 성공하면 사용자는 인증되는 것이며 두 인증 모두 실패한 경우, 어플리케이션 액세스는 거부된다.</li> </ul>
네임스페이스	이 어플리케이션의 페이지들이 실행되는 Caché 네임스페이스

필드	설명
Caché 파일 율리적 경로	CSP 소스 파일이 저장된 Caché 서버의 디렉터리이며 이 경로는 Caché 서버 시스템의 install-dir/csp/ 디렉터리내에 있다.

Recurse	어플리케이션의 물리적 디렉토리의 하위 디렉터리도 액세스 가능한지의 여부를 규정한다. UPath가 URL 경로이고 PPath가 물리적인 경로라고 한다면, Recurse가 예로 설정된 경우 UPath/xxx/yyy 는 PPath/xxx/yyy 에서 CSP 파일을 찾는다. Recurse가 아니오로 설정된 경우에는 UPath에 직접 포함된 파일만 사용된다.
자동 컴파일	CSP 서버가 CSP 소스 파일을 자동으로 컴파일할 것인지의 여부를 규정한다. 이 필드가 예로 설정된 경우, CSP 파일이 컴파일된 클래스보다 최신이면 다시 컴파일한다. 일반적으로 개발 환경에서는 예로, 서비스 환경에서는 아니오로 구성한다.
이벤트 클래스	CSP 어플리케이션 이벤트(예: 타임아웃 또는 세션 종료)가 발생할 때 호출되어야 하는 메소드들을 포함하고 있는 CSP 클래스(%CSP.SessionEvents의 하위 클래스)의 디폴트 이름을 설정한다. 입력된 값은 %CSP.Session 오브젝트의 EventClass 프로퍼티값으로 대체할 수도 있다.
디폴트 타임아웃	디폴트 세션 타임아웃(초). %CSP.Session 오브젝트의 AppTimeout 프로퍼티값으로 대체 가능하다.
디폴트 상위클래스	CSP 파일로부터 생성되는 클래스들의 디폴트 상위클래스로서 CSP 컴파일러에 의해 사용되며 디폴트 클래스는 %CSP.Page 이다.
세션에 대한 쿠키 사용	CSP로 하여금 쿠키 또는 URL 재작성 기법(각 URL에 값을 저장)을 사용하여 현재 브라우저에서 사용되는 세션을 추적하고자 하는 여부. (이 옵션과 상관없이 어플리케이션으로 하여금 실제로 쿠키를 사용하게 하기 위해서는 어플리케이션이 쿠키를 사용하도록 작성하면 된다). 선택값으로는 a) (항상) 쿠키 사용, b) 쿠키 사용 안함(금지), c) (디폴트값) 클라이언트 브라우저가 쿠키를 비활성화시키지 않은 한 쿠키 사용(자동인식) 등이 있다. 이 옵션은 어플리케이션의 쿠키 사용 여부를 결정하는 값이 아니다 (쿠키 사용 여부는 어플리케이션을 어떻게 작성 하느냐에 따라 결정된다). 즉, CSP가 세션을 관리하는 방식만을 제어한다. 사용자가 쿠키를 사용하지 않는 경우, 어플리케이션은 URL 재작성을 사용한다.

필드	설명
세션 쿠키 경로	세션 쿠키의 범위. 이 옵션은 세션 쿠키를 Caché로 전송하기 위해 브라우저가 사용하는 URL을 설정한다. 예를 들어, 어플리케이션명이 myapp인 경우, 디폴트 값 /myapp/ 의 의미는 /myapp/ 내의 모든 페이지들에 대한 쿠키들만을 전송한다는 의미이다. 이를 어플리케이션에 의해 요청되는 것으로만 제한하는 경우, 동일 시스템상의 다른 CSP 어플리케이션들이나 동일 웹 서버상의 다른 어플리케이션에 의해 이 세션 쿠키가 사용되어지는 것을 막을 수 있다. 그리고 브라우저 및 쿠키는 대소문자를 구별하기 때문에 세션 쿠키를 '/'로 설정하면 어플리케이션명의 대소문자가 바뀌더라도 야기될 수 있는 라이센스 또는 세션 문제를 방지할 수 있다.
파일 처리	<p><b>아니오</b> - 이 어플리케이션 경로에서 파일을 처리하지 않는다.</p> <p><b>항상</b> - 디폴트. 이 어플리케이션에서 항상 파일을 처리하며 정적 파일을 위한 이 경로에 대한 CSP 보안 설정은 무시한다. 이 값은 웹 서버로부터 처리되는 파일들과 역방향으로 호환되기 때문에 새로운 어플리케이션에 대한 기본 값이다.</p> <p><b>항상 그리고 캐싱</b> - 이 어플리케이션로부터의 파일들은 항상 처리되며 Cache 서버로의 반복적 요청을 피하기 위하여 CSP Gateway로 하여금 처리 파일들을 캐싱할 수 있도록 한다. 이 모드는 구현된 어플리케이션이 사용할 것으로 예상하는 모드이다.</p> <p><b>CSP 보안 사용</b> - 이 어플리케이션에서 csp/cls 페이지를 볼 수 있는 권한이 있는 경우, 정적 파일 또한 볼 수 있다. csp/cls 페이지를 볼 수 있는 권한이 없는 경우 404 page not found 오류가 발생한다.</p>
파일 처리 타임아웃	정적 파일이 브라우저에 의해 액세스되어져야 하는 최대 시간(초). 디폴트 값은 3600 초이다.
CSP 이름 Lock	'예' 인 경우, 두 CSP 어플리케이션 모두 동일한 네임스페이스와 패키지에 액세스했을 때 먼저 파일을 로드한 어플리케이션을 통해서만 해당 CSP 페이지를 액세스할 수 있다.
사용자 페이지 오류	어플리케이션에서 페이지를 생성할 때 오류가 발생하는 경우 실행되는 .csp 또는 .cls 의 확장자를 갖는 페이지 이름
패키지명	CSP 컴파일러에 의해 사용되기 위해서 패키지명 앞에 첨부되는 이름. 이 이름은 CSP 파일로부터 생성되는 클래스에 사용된 패키지 이름에 사전 추가된다. 이 필드를 설정하지 않으면 디폴트 값 csp 가 사용된다.

필드	설명
로그인 페이지	사용자 정의 로그인 페이지 이름. /csp/user/application/mylogin.cps, mylogin.csp, /cspuser/application/mylogin.cls, application.mylogin.cls 등과 같은 형식으로 정의할 수 있다. CSPSystem 사용자는 반드시 %DB_User 또는 %All 역할을 가져야 한다.
패스워드 변경 페이지	패스워드 변경 시 사용할 페이지명.

**응용프로그램 Role** 탭을 사용하여 해당 어플리케이션을 사용하는 동안 사용자에게 할당될 역할을 선택한다. 여기서 선택한 응용프로그램 Role은 사용자에게 이미 할당된 역할들에 추가된다.

**일치하는 Role** 탭을 사용하여 현재의 Role 지정을 기반으로 어플리케이션 사용시 어플리케이션 사용자를 추가적 역할에 지정한다.

Role에 대한 보다 자세한 설명은 온라인 문서 *Caché Security Administration Guide* 의 “Applications”을 참조하기 바란다.

## 새로운 어플리케이션 정의

CSP 서버에 /myapp 라는 새로운 CSP 어플리케이션을 다음 절차에 따라 정의 한다:

1. 시스템 관리 포탈에서 ([홈] > [보안관리] > [CSP 응용프로그램]) 페이지로 이동한 후 ‘CSP 응용프로그램 만들기’를 클릭한다.
2. 새로운 어플리케이션 이름을 CSP 응용프로그램명에 입력한다(이 경우, /myapp).
3. 필요한 어플리케이션 프로퍼티들을 입력한다(대부분의 경우, 선택적이다). 중요한 항목들은 다음과 같다:
  - 허용된 인증 메소드 — 어플리케이션에 연결하기 위한 유효한 인증 방법
  - 네임스페이스 — 어플리케이션이 실행되는 Caché 네임스페이스
  - CSP 파일 물리적 경로 — CSP 파일의 실제 저장 위치 (HTML 기반 개발 사용 시)
4. ‘저장’ 버튼을 클릭한다.

5. **응용프로그램 Role** 탭을 선택하여 어플리케이션 실행시 필요한 사용자 역할을 선택한다. 이 응용프로그램 Role을 이미 사용자에게 할당되어 있는 Role들에 추가된다.
6. **일치하는 Role** 탭에서 현재의 역할 지정을 기반으로한 어플리케이션 실행시 어플리케이션 사용자를 추가적으로 역할에 할당한다.

# 제 8 장 Caché에서 SOAP과 웹 서비스 사용하기

## 1. 소개

- 1) Caché의 웹 서비스 지원
- 2) Caché의 웹 클라이언트 지원

## 2. 웹 서비스 생성

- 1) Caché 웹 서비스 개요
- 2) 기본 필수 사항
- 3) 카타로그 및 테스트 페이지
- 4) 웹 서비스의 서비스명과 네임스페이스명
- 5) 웹 서비스에서 지원되는 SOAP 버전
- 6) WSDL 보기

## 3. 웹 메소드 생성

- 1) 기본 필수 사항
- 2) 참조 또는 Output 인수로 값 리턴
- 3) Input/Output 인수값으로 특수문자 사용
- 4) 오브젝트를 Input/Output 인수로 사용
- 5) 컬렉션을 Input/Output 인수로 사용
- 6) 데이터 집합을 Input/Output 인수로 사용
- 7) 클래스 쿼리를 웹 메소드로 사용하기
- 8) SOAP 메시지에 대한 바인딩 스타일 명사하기
- 9) SOAP 메시지 인코딩 설정

## 4. 웹 클라이언트 생성

- 1) Caché SOAP 클라이언트 마법사 개요
- 2) SOAP 클라이언트 마법사 사용하기

## 1. 소개

Caché는 SOAP(Simple Object Access Protocol) 1.1 과 1.2 를 지원한다. 따라서, Caché는 SOAP 규격과의 완벽한 호환성으로 갖춘 사용하기 쉽고 효율적 지원을 제공한다. 또한 이러한 기능이 Caché에 내장되어 있기 때문에, 별도의 복잡한 미들웨어나 운영 시스템 확장이 필요 없으며, Caché가 지원하는 모든 플랫폼에서 사용할 수 있다.

다음은 Caché의 SOAP 를 활용하여 수행할 수 있는 업무들이다:

- 웹 서비스 정의와 계시: SOAP 프로토콜을 사용하여 클라이언트 어플리케이션에서 호출하는 메소드들의 컬렉션. 이 메소드들은 다른 SOAP 호환 응용프로그램에서도 검색하여 호출할 수 있다. Caché는 SOAP 메소드를 데이터베이스 내에서 직접 실행하므로 효율성이 매우 높다.
- 기존의 WSDL 문서를 사용하여, 그 WSDL에 기술된 웹 서비스를 호출할 수 있는 Caché 웹 클라이언트를 생성한다.

본 장에서는 다음 항목에 대해 설명한다.

- Caché의 웹 서비스 지원
- Caché의 웹 클라이언트 지원
- Caché의 웹 서비스와 SOAP 지원에 따른 추가적인 기능
- WSDL 문서에 대한 제반 사항을 포함하여 Caché가 지원하는 표준

### 1) Caché의 웹 서비스 지원

Caché에는 웹 서비스에 대한 효율적인 빌트-인 지원 시스템이 있다. 기존의 클래스들은 약간의 변경만으로도 웹 서비스로 전환할 수 있으며, 처음부터 새로운 웹 서비스를 만들 수도 있다. 어떤 경우든 다음 내용들이 공통적으로 적용된다:

- 웹 서비스 클래스는 %SOAP.WebService 클래스로부터 상속한다.
- 메소드나 클래스 쿼리를 WebMethod 키워드로 플래그하면, 웹 메소드로서 호출할 수 있다.
- 웹 메소드의 모든 입력과 출력은 SOAP 메시지로 사용될 수 있도록, XML로 전환되어져야 한다. 일반 리터럴(literal) 데이터에 대해서는 추가 작업이 필요 없지만 입출력으로 사용되는 오브젝트의 경우 모두 XML로 전환 가능해야 한다 - 즉 해당 오브젝트는 %XML.Adaptor 에서 상속되어져야 한다. 한편 데이터 집합은 %XML.DataSet 에서 상속해야 한다.

다음은 Caché 웹 서비스에 의해 수행되는 작업들이다:

- 착신 SOAP 메시지 검증.
- SOAP 메시지를 Caché 표현으로 전환하고, 대응하는 메소드 호출.

- 메소드를 Caché 데이터베이스에서 직접 실행; 매우 높은 효율성 구현.
- 응답 메시지를(SOAP 메시지) 호출자에게 반환.

또한 %SOAP.WebService 클래스가 %CSP.Page를 상속하기 때문에, Caché 웹 서비스도 CSP 어플리케이션이며 따라서 CSP (Caché Server Pages) 기술의 장점들을 공유한다. 여기서 CSP 응용프로그램이란 하나의 논리적 응용프로그램 이름과 연결되어 있는 여러 CSP 페이지들의 집합으로서 CSP 게이트웨이에 의해 하나의 단위로 관리된다.

## 웹 서비스를 위해 생성되는 도구

Caché 클래스 컴파일러는 웹 서비스를 위한 WSDL을 생성하고 (CSP를 사용하는) 웹 서버를 통하여 이를 게시한다. 이 WSDL은 WS-I (Web Services Interoperability Organization)에 의해 제정된 Basic Profile 1.0에 따라 컴파일된다. WSDL 문서는 동적으로 제공되며, 웹 서비스 클래스의 인터페이스가 변경되면 해당 변경이 자동적으로 반영된다. 따라서 WSDL 문서를 직접 관리할 필요는 없다.

또한 웹 서비스 컴파일시, Caché는 테스트를 위해 웹 서비스의 카탈로그 페이지를 생성하는데, 이 페이지에는 서비스에 정의되어 있는 메소드들의 목록과 WSDL 문서와의 링크가 포함되어 있다.

## HTTP 인터페이스

Caché 웹 서비스에서는 HTTP에 대해 두 가지 인터페이스를 지원한다:

- SOAP, SOAPAction HTTP 헤더, SOAP <Envelope> 와 <Body> 엘리먼트 등을 사용.
- CSP 페이지 (제한된 테스트 용도로만 사용). 메소드 이름과 인수를 쿼리 파라미터로 취한다. 이 테스트 페이지는 SOAP 경로 전체를 테스트하지는 않는다. 테스트 페이지들은 위에서 언급된 카탈로그 페이지에 포함되어 있다.

XML 형식의 응답 메시지는 위의 두 인터페이스에서 동일하다.

## Caché 웹 서비스 아키텍처

Caché 웹 서비스는 CSP 응용프로그램의 일부로서, 웹 서비스가 실행되기 위해서는 반드시 Caché 가 운영중이어야 한다.

Caché 웹 서비스가 기본적으로 어떻게 작동되는지를 이해하려면, 웹 서비스가 메시지(하나의 SOAP 메시지를 갖는 단일 HTTP 요청)를 받을 때 발생하는 이벤트들을 순서대로 따라가 보는 것이 도움이 될 것이다.

우선 특정 URL로 나타나는 HTTP 요청의 내용을 리뷰해 보자:

- HTTP 헤더: HTTP 버전, 문자 집합, 기타 관계 정보를 표시한다. HTTP 헤더는 HTTP를 통해 웹 메소드를 호출할 때 필요한 **SOAPAction** HTTP 요청 헤더 필드를 포함하는데 이는 SOAP HTTP 요청의 대상을 의미하는 URI이다. SOAP에서는 URI의 형식이나 속성에 어떠한 제약도 두지 않는다.

- 요청 라인: GET, POST, HEAD 와 같은 HTTP 메소드를 포함하여 실행할 액션을 나타낸다.
- 메시지 본문: 메소드 호출을 담고 있는 SOAP 메시지를 말한다. SOAP 메시지는 호출할 메소드 이름과 그 인수로 사용할 값을 나타낸다. 또한 이 메시지에 SOAP 헤더를 포함할 수도 있다.

다음은 요청이 전송되었을 때의 수행되는 액션들이다:

1. 웹 서버에 요청이 전달된다.
2. 요청된 URL이 확장자 .cls 로 끝나는 경우, 웹 서버는 그 요청을 CSP 게이트웨이로 전송한다.
3. CSP 게이트웨이는 URL을 리뷰하고 URL의 해당 부분을 CSP 응용프로그램의 논리적 이름으로 변경한 후 CSP 응용프로그램내의 대응하는 물리적 위치로 전송한다.
4. 웹 서비스 페이지가 요청을 받으면, OnPage 메소드가 호출된다.
5. 웹 서비스는 요청에 Caché SOAP 세션 헤더가 포함되어 있는지를 검사하여, 있으면 해당 SOAP 세션을 재사용하고, 없으면 새로운 세션을 시작한다.

**참고:** 이 단계는 Caché SOAP 속성에 따라 지원되는 SOAP 세션을 의미한다. SOAP 규격 자체는 세션에 대한 표준을 정의하고 있지 않는다. 그러나 Caché SOAP 지원에서는 웹 클라이언트와 웹 서비스간에 세션을 유지하는데 사용되는 전용 Caché SOAP 세션 헤더를 제공하고 있다.

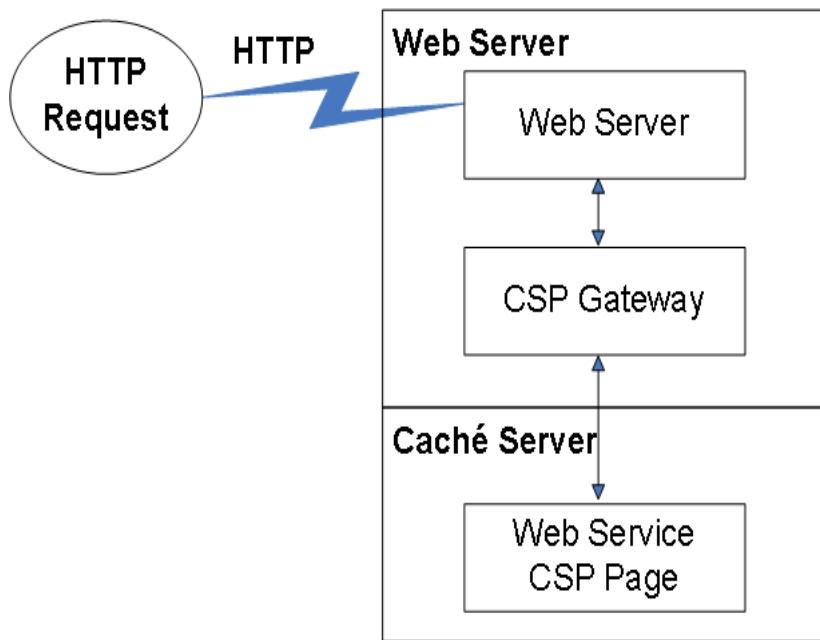
6. 웹 서비스는 메시지의 정당성 검사를 하고, 모든 입력 파라미터를 대응하는 Caché 형식으로 전환한다. 변환 과정에서 복합 타입의 파라미터에 대해서는, 해당 유형에 대응하는 오브젝트 인스턴스를 생성하고 이 오브젝트를 웹 메소드의 입력으로 사용한다.

SOAPAction 헤더는 메소드와 그에 따른 요청 오브젝트를 결정하는 데 사용한다.

웹 서비스가 메시지를 리뷰할 때, 새로운 요청 오브젝트를 생성하여, SOAP 메시지를 이 오브젝트에 임포트(import)한다. 이 과정에서, 웹 서비스에서는 웹 서비스를 컴파일 할 때 생성한 클래스(웹 메소드 취급 클래스)가 사용된다.

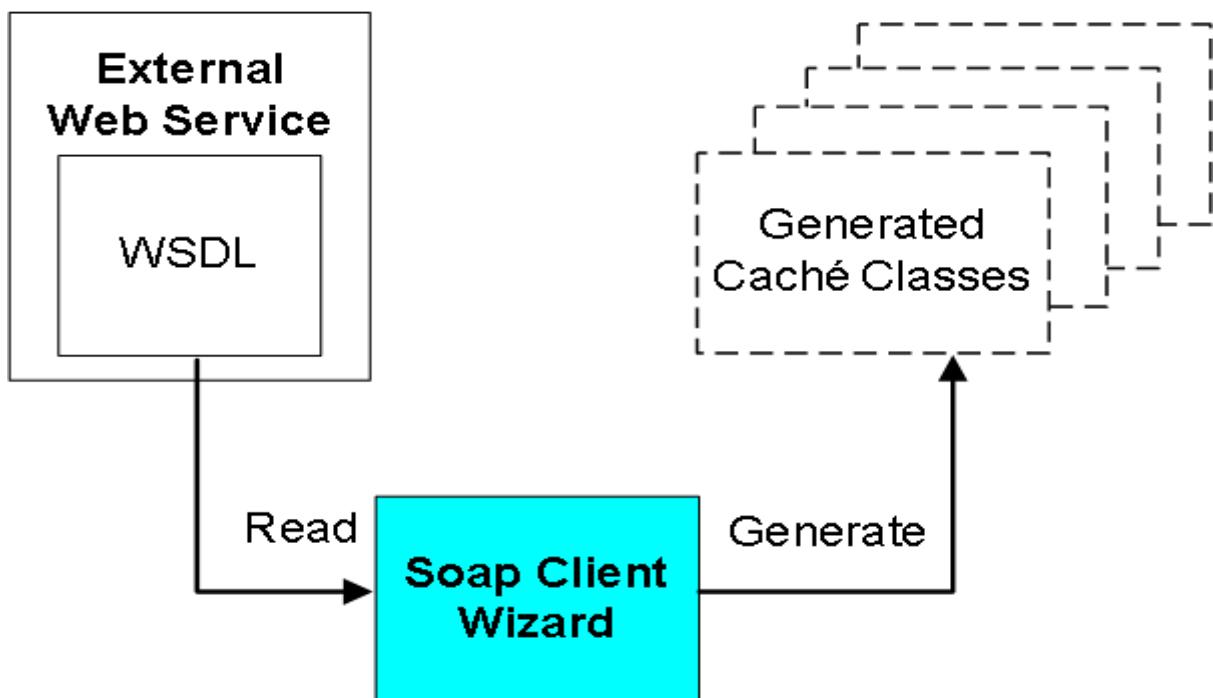
7. 웹 서비스는 요청된 Caché 메소드를 실행하고, 결과를 구성하며, 필요에 따라 SOAP 헤더를 포함한 SOAP 응답을 만든다.
8. 웹 서비스는 SOAP 응답(XML 문서)을 현재의 출력 디바이스에 출력한다.

다음 그림은 이러한 절차의 외형적 절차 부분을 보여준다.

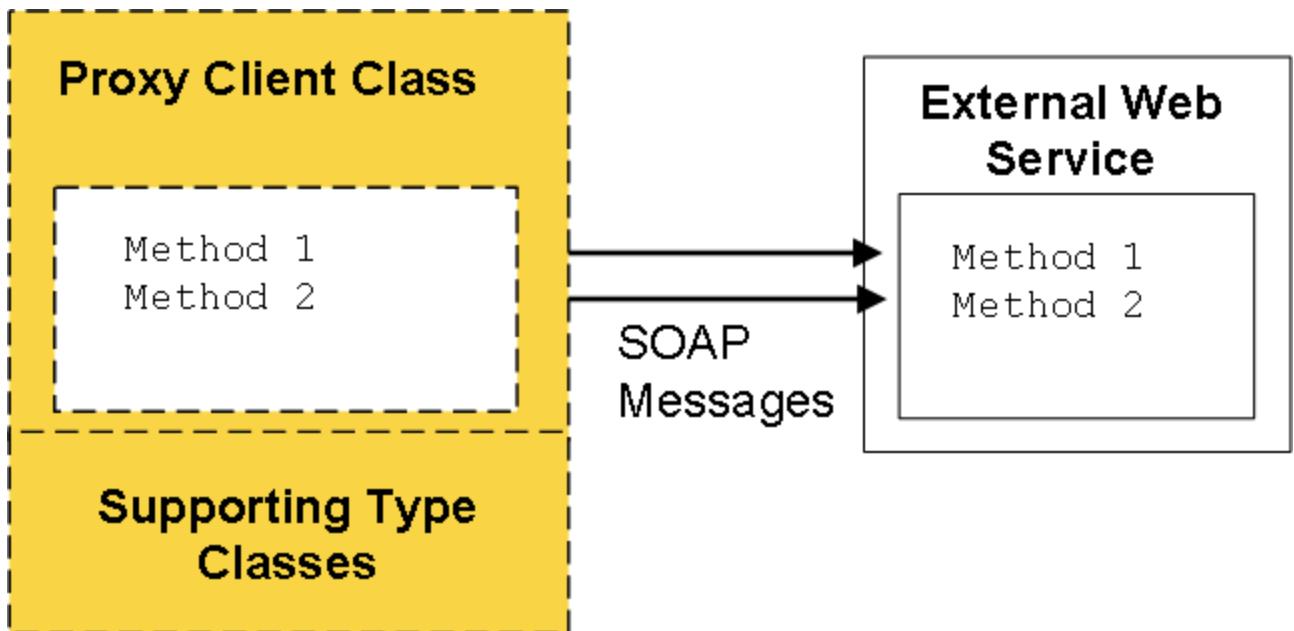


## 2) Caché의 웹 클라이언트 지원

Caché 환경에서는 거의 모든 형태의 웹 서비스에 대한 웹 클라이언트를 쉽게 생성할 수 있다. 웹 클라이언트 작성은 Caché SOAP 웹 클라이언트 마법사를 사용하는데, 이 마법사를 통해 WSDL 문서를 읽고 웹 클라이언트로서 사용할 Caché 클래스들의 집합을 생성한다. 이 마법사는 Studio의 마법사 메뉴를 통해 사용할 수도 있고, 프로그램적으로 메소드를 호출하여 사용할 수 있다.



생성된 웹 클라이언트 인터페이스에는, 웹 서비스에 의해 정의된 각 메소드들에 대한 프록시 메소드를 가지는 클라이언트 클래스가 포함되어 있다. 각 프록시 클래스는 상응하는 웹 서비스 메소드에서 사용하는 동일한 서명을 사용한다. 인터페이스 또한 메소드의 입력/출력 인수로서 필요한 XML 타입을 정의한 클래스들을 포함한다.



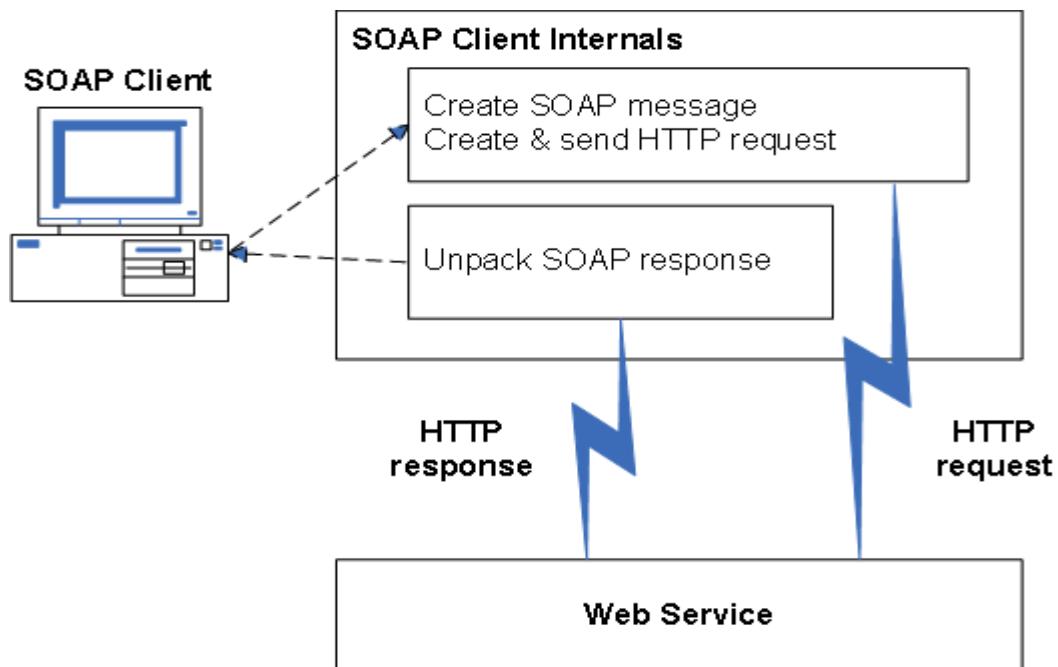
일반적으로 생성된 클래스를 임의로 재편집하지 않으며, 필요한 경우 사용자의 웹 클라이언트 기능을 제어하고 이 클라이언트의 프록시 메소드를 호출하는 클래스를 추가로 생성한다.

#### Caché 웹 클라이언트 아키텍처

Cache 웹 클라이언트가 어떻게 실행되는지를 이해하기 위해 다음과 같이 웹 클라이언트내의 메소드가 호출할 때, 발생하는 이벤트를 따라가 본다:

1. 웹 클라이언트는 호출 메소드와 필요한 인수 값들로 구성된 SOAP 메시지를 생성한다.
2. 이 SOAP 메시지를 포함하는 HTTP 요청을 생성한다. HTTP 요청은 위에서 설명한 바와 같이, 요청 라인과 HTTP 헤더를 포함한다.
3. HTTP 요청을 해당 URL에 보낸다.
4. HTTP 응답을 기다리고 상태를 체크한다.
5. 웹 서비스로부터 SOAP 응답을 받는다.
6. SOAP 응답을 출력한다.

다음은 위 절차의 전체적인 그림을 보여준다.



## 2. 웹 서비스 생성

이 장에서는 Caché 에서 웹 서비스를 생성하는 방법을 설명한다:

- Caché 웹 서비스 개요
- 웹 서비스와 웹 메소드를 생성하는 데 필요한 기본 필수 사항
- 웹 서비스용 Caché 카탈로그 및 테스트 페이지 소개
- 웹 서비스에 사용하는 서비스 이름과 네임스페이스 설정 방법
- WSDL에서 주어진 SOAP 버전 설정 방법
- 웹 서비스를 위해 생성된 Caché WSDL 보는 방법
- 웹 서비스를 테스트하는 데 사용하는 도구

### 1) Caché 웹 서비스 개요

Caché에서 웹 서비스를 생성하려면, %SOAP.WebService를 상속하는 클래스를 만들어야 하는데, 이 클래스는 SOAP 프로토콜을 사용하여 호출되는 메소드들에 필요한 모든 기능을 제공할 뿐만 아니라 서비스를 기술하는 WSDL 문서의 관리등과 같은 SOAP 관련 문서 관리 작업을 자동적으로 수행한다.

또한 이 클래스는 %CSP.Page 클래스도 상속하게 되기 때문에 모든 웹 서비스 클래스는 (CSP를 통해) HTTP 요청에 응답할 수 있다. 따라서 %SOAP.WebService 클래스는 다음 작업들을 수행하는 HTTP

이벤트에 대응하는 메소드들을 실행한다:

- 웹 서비스 및 해당 메소드를 기술한 HTML 카타로그 페이지 게시. 페이지에 대한 설명은 클래스 정의에 포함되어 있는 주석으로 표시된다.
- 웹 서비스용 WSDL 문서를 XML 문서로 게시한다.

## 2) 기본 필수 사항

Caché에서 웹 서비스를 생성, 재제작하려면 다음의 필수 사항을 준수하는 Caché 클래스를 생성하고 컴파일 한다:

- 클래스는 반드시 %SOAP.WebService에서 상속되어야 한다.
- 클래스는 **SERVICENAME** 파라미터를 정의해야 한다. 이 파라미터가 정의되어 있지 않으면 클래스는 컴파일되지 않는다.
- 웹 메소드로 사용하기 위해하기 위해서는 다음과 같이 메소드를 정의해야 한다:
  - 메소드에 WebMethod 키워드가 설정되어져야 된다.
  - 오브젝트를 인수 또는 리턴 값으로 사용하려면, 해당 오브젝트는 반드시 XML 구성 가능하여야 한다. 즉, 클래스 정의는 %XML.Adaptor를 상속하여야 한다. 일반적으로 이 클래스에 대한 기본 설정이면 충분하지만 추가 설정이 필요한 경우 온라인 문서 “Using XML with Caché”를 참조하기 바란다.
  - 데이터 집합을 인수 또는 리턴값으로 사용하려면, 데이터 집합의 타입이 %XML.DataSet(또는 그 하위 클래스)이어야 한다. %XML.DataSet은 표준 %ResultSet의 XML 가능 하위 클래스이다.
  - 컬렉션(%ListOfObjects 또는 %ArrayOfObjects)을 인수 또는 리턴값으로 사용하려면, 컬렉션의 ELEMENTTYPE 파라미터를 XML 가능 클래스로 설정하여야 한다.
- 웹 메소드로 사용할 수 있는 클래스 쿼리를 정의할 수도 있는데, 이 경우 해당 쿼리는 WebMethod 키워드로 설정되어져야 한다.

## Studio 사용하여 새로운 웹 서비스 생성하기

Caché Studio에서 새로운 웹 서비스 클래스를 생성하려면, 아래와 같은 절차로 마법사를 사용한다:

1. Studio를 열고 연결된 Caché 네임스페이스가 정확한 작업 네임스페이스인지를 확인한다. 아닌 경우, 파일 > 네임스페이스 변경을 사용하여 원하는 네임스페이스로 변경한다.
2. 메뉴 파일 > 새로 만들기를 클릭하여 ‘새로 만들기’ 대화 상자를 오픈한다.
3. 사용자 정의 탭을 선택한다.
4. 웹 서비스 아이콘을 선택하고 확인 버튼을 클릭하면 Caché 웹 서비스 마법사창이 오픈된다.

5. 필수 항목 패키지명, 클래스명, 서비스명에 원하는 값을 입력한다.
6. 필요한 경우, 서비스 네임스페이스 URI 값을 변경한다. (이 초기 값은 후에 변경 가능하다.) 이 값은 Caché 네임스페이스가 아닌 XML 네임스페이스이다.
7. ‘서비스 메소드들’ 입력란 각 줄에 메소드명들을 입력하여도 된다.(메소드들은 나중에 Studio에서 해당 서비스 클래스 정의를 편집할 때 하나씩 추가할 수 있다.)
8. 확인을 클릭한다.

이제 웹 메소드들에 대한 정의들을 포함하고 있는 새로운 웹 서비스 클래스가 다음과 같이 생성되었다:

```
/// MyApp.StockService
Class MyApp.StockService Extends %SOAP.WebService [ ProcedureBlock ]
{
  /// 웹서비스명.
  Parameter SERVICENAME = "StockService";

  /// TODO: 이것을 실제 SOAP 네임스페이스로 변경.
  /// 웹서비스를 위한 SOAP 네임스페이스
  Parameter NAMESPACE = "http://tempuri.org";

  /// 참조된 클래스들의 네임스페이스들이 WSDL에서 사용될 것입니다.
  Parameter USECLASSNAMESPACES = 1;

  /// TODO: 인수 및 실행 추가.
  /// Forcast
  Method Forecast() As %String [ WebMethod ]
  {
    Quit "Forecast"
  }
}
```

예제: 다음의 예처럼, 정의된 웹 메소드에 추가로 원하는 작업을 수행한다:

```
/// MyApp.StockService
Class MyApp.StockService Extends %SOAP.WebService [ ProcedureBlock ]
{
  /// 웹서비스명.
  Parameter SERVICENAME = "StockService";

  /// TODO: 이것을 실제 SOAP 네임스페이스로 변경.
  /// 웹서비스를 위한 SOAP 네임스페이스
  Parameter NAMESPACE = "http://tempuri.org";

  /// 참조된 클래스들의 네임스페이스들이 WSDL에서 사용될 것입니다.
  Parameter USECLASSNAMESPACES = 1;
```

```

/// TODO: 인수 및 실행 추가.
/// Forecast
Method Forecast(StockName As %String) As %String [ WebMethod ]
{
    Set price = $Random(1000)
    Quit price
}

```

### 3) 카타로그 및 테스트 페이지

웹 서비스 클래스를 컴파일하면, 클래스 컴파일러는 웹 서비스를 검사하고 해당 서비스의 WSDL을 보는데 사용되는 카타로그 페이지를 생성한다. 이 CSP 페이지를 보려면,

1. Studio의 편집창에 해당 웹 서비스 클래스가 오픈되어 있음을 확인 한다.
2. 메뉴 보기 > 웹 페이지를 선택하면 아래와 같은 형식의 URL로 만들어진 카타로그 페이지가 브라우저에 나타난다.

base/csp/app/web\_serv.cls

여기에서, **base** 는 (포트번호를 포함한) 사용자 웹 서버의 기본 URL이며, **/csp/app** 는 웹 서비스가 포함되어 있는 CSP 응용프로그램의 이름이며, **web\_serv** 는 웹 서비스의 클래스 이름이다(일반적으로 /csp/app 는 /csp/namespace 이다). 예를 들면, 다음과 같은 URL 이 요청된다:

<http://localhost:57772/csp/samples/MyApp.StockService.cls>

카타로그 페이지는 다음과 같이 클래스명, 네임스페이스, 서비스명 뿐만 아니라 클래스 및 웹 메소드에 대한 주석도 보여준다:

Web Service MyApp.StockService

MyApp.StockService

This web service is using <http://tempuri.org> as its default namespace.

The name of this web service is **StockService**

The following operations are supported. For a formal definition, please review the [Service Description](#)

- **Forecast**  
This method returns tomorrow's price for the requested stock

Service Description 링크는 WSDL 문서를 브라우저에 출력한다. WSDL 문서에 대한 추가 정보는 이 장

후반부의 “WSDL 보기”를 참조한다. **Forecast** 링크는 예제와 같이 연계된 메소드(예, Forecast)를 테스트할 수 있는 페이지로 이동한다:

웹 서비스 MyApp.MyService

---

## Forecast

TODO: 인수 및 실행 추가. Forcast

**테스트**

HTTP GET 프로토콜을 사용하는 작업을 테스트하려면 '**호출**' 버튼을 클릭하십시오.

파라미터	값
StockName	<input type="text"/>
	<b>호출</b>

위의 페이지에서 해당 웹 서비스를 테스트할 경우 다음과 같은 제한이 따른다:

- SOAP 요청 자체를 볼 수는 없다.
- 전체 SOAP 경로를 테스트 하지는 않는다. 그 의미는 SOAP 경로에 기반한 로그를 남기지 않는다는 것이다.
- 간단한 텍스트 타입의 데이터만 입력 가능하며, 오브젝트나 데이터 집합과 같은 타입의 인수를 사용하여 메소드를 호출할 수는 없다.

## 4) 웹 서비스의 서비스명과 네임스페이스명

웹 서비스 클래스에서는 웹 서비스를 위한 다음과 같은 파라미터들을 정의하여야 한다.

### 웹 서비스 속성

항목	설명
NAMESPACE	웹 서비스의 대상 네임스페이스를 정의한 URL - 해당 서비스와 컨텐츠가 다른 서비스와의 충돌 방지. 초기값은 <a href="http://tempuri.org">http://tempuri.org</a> 로 SOAP 개발 시 사용하는 임시 URL이다. 이 값을 정의하지 않으면, 타깃 네임스페이스는 <a href="http://tempuri.cog">http://tempuri.cog</a> 가 된다.
RESPONSENAMESPACE	응답 메시지의 네임스페이스를 정의하는 URI. 디폴트로, NAMESPACE 파라미터값과 동일하다.
TYPENAMESPACE	웹 서비스에서 정의되는 스키마 타입에 대한 네임스페이스. 이 파라미터를 정의하지 않으면, 스키마는 웹 서비스의 타깃 네임스페이스에 속하게 된다 (NAMESPACE의 값 또는 디폴트 <a href="http://tempuri.org">http://tempuri.org</a> ).
RESPONSETYPENAMESPACE	응답 메시지가 사용하는 타입들에 대한 네임스페이스를 정의하는 URI. 디폴트로, TYPENAMESPACE 파라미터값과 동일하다.

SERVICENAME	웹 서비스명. 이름은 문자로 시작해야 하며, 문자와 숫자로만 이루어져야 한다. 이 파라미터의 값이 설정되지 않으면, 웹 서비스는 컴파일되지 않는다.
-------------	--

## 5) 웹 서비스에서 지원되는 SOAP 버전

본 버전 출간 시점에서는, Caché Web 웹 서비스는 SOAP 1.1 과 1.2 를 지원하고 있다. 웹 서비스의 WDSL에서 선전되는 SOAP 버전을 명시하려면, 다음과 같이 파라미터를 명시하면 된다.

항목	설명
SOAPVERSION 파라미터	<p>이 웹 서비스를 사용하는 모든 SOAP 요청에서 필요한 SOAP 버전을 명시한다 :</p> <ul style="list-style-type: none"> <li>"" : SOAP 1.1 또는 1.2.</li> <li>"1.1" : SOAP 1.1. 디폴트 값</li> <li>"1.2" : SOAP 1.2.</li> </ul>

웹 서비스가 SOAP 요청을 받으면, 웹 서비스의 *SoapVersion* 프로퍼티값이 요청된 SOAP 버전으로 업데이트 된다. 만약 지원되지 않는 SOAP 버전을 사용하는 경우, Misunderstood 오류를 반환한다.

## 6) WSDL 보기

웹 서비스를 정의하기 위해 %SOAP.WebService를 사용할 때, Caché는 해당 웹 서비스를 설명하는 WSDL 문서를 작성한다. 그리고 웹 서비스를 수정, 컴파일 할 때마다, WSDL 문서는 자동 갱신된다. 이 섹션에서는 다음 항목에 대해 설명한다 :

- WSDL과 WSDL이 게시되는 URL 보기
- WSDL을 정적 문서로 생성하는 데 사용하는 메소드

### WSDL 보기

웹 서비스에 대한 WSDL을 보려면 다음과 같이 URL을 사용한다:

```
base/csp/app/web_serv.cls?WSDL
```

여기에서, **base** 는 웹 서버의 기본 URL(필요하면 포트 포함)이며, **/csp/app** 는 웹 서비스가 속한 CSP 응용프로그램의 이름, **web\_serv** 는 웹 서비스 클래스명이다. (일반적으로, **/csp/app** 는 **/csp/namespace** 이다.)

다음 예은 포트번호 57772 를 사용하는 localhost 웹 서버에 CSP 응용프로그램 /csp/samples 에 속한 웹 서비스 MyApp.StockService 의 WSDL에 대한 요청 URL 이다:

```
http://localhost:57772/csp/samples/MyApp.StockService.cls?WSDL
```

참고: 클래스 이름에 있는 퍼센트 문자(%) 가 있는 경우, URL에서 밑줄 문자로(\_) 대체된다.

또는 위에서 설명한 **Service Description** 링크를 통해 WSDL 문서를 볼 수도 있다. 다음은 XML 형식으로 출력된 WSDL 문서의 예이다:

```
<?xml version="1.0" encoding="UTF-8" ?>
- <definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:SOAP-
  ENC="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
  xmlns:s="http://www.w3.org/2001/XMLSchema" xmlns:s0="http://tempuri.org"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  targetNamespace="http://tempuri.org">
- <types>
- <s:schema elementFormDefault="qualified" targetNamespace="http://tempuri.org">
- <s:element name="Forecast">
- <s:complexType>
- <s:sequence>
<s:element minOccurs="0" name="StockName" type="s:string" />
</s:sequence>
</s:complexType>
```

## WSDL 생성

또한 WSDL은 정적 문서로 생성할 수 있는데, 이를 위해 %SOAP.WebService 클래스는 두 가지 메소드를 제공한다.

- **FileWSDL()** WSDL 문서를 주어진 파일로 내보낸다. 인수로 파일명을 받는다.  
참고: 이 메소드를 사용하기 전에 반드시 웹 서비스의 *LOCATION* 파라미터를 정의해야 한다. 그렇지 않으면 WSDL에는 서비스 주소가 포함되지 않는다.
- **GenerateWSDL()** WSDL을 현재 설정되어 있는 디바이스로 출력한다. 이 메소드에는 인수가 없다.

두 메소드 모두 값을 반환하지 않는다.

### 3. 웹 메소드 생성

#### 1) 기본 필수 사항

이 섹션에서는 웹 서비스 클래스에 포함되는 웹 메소드를 정의하는데 필요한 사항에 대해 설명한다.

- 인스턴스 메소드 또는 클래스 메소드를 웹 메소드로 사용하려면,
  - 메소드를 WebMethod 키워드로 설정한다.
  - 메소드가 오브젝트를 인수나 리턴 값으로 사용하는 경우, 그 오브젝트는 XML 구성 가능하여야 한다. 즉, 해당 오브젝트의 클래스 정의가 %XML.Adaptor를 상속하고 있어야 한다.
  - 메소드가 데이터 집합을 인수나 리턴 값으로 사용하는 경우, 데이터 집합의 타입은 %XML.DataSet – 표준 %ResultSet 의 XML 구성 가능 하위클래스 – 이어야 한다.
  - 컬렉션(%ListOfObjects 또는 %ArrayOfObject)을 인수나 리턴 값으로 사용하는 경우, 컬렉션의 ELEMENTTYPE 파라미터가 XML 구성 가능 클래스로 설정되어야 한다.
- 클래스 쿼리를 웹 메소드로 사용하려면, 해당 쿼리에 WebMethod 키워드를 설정한다.

**참조.** 웹 서비스는 해당 웹 서비스 클래스에서 웹 메소드로 정의된 메소드들만으로 구성된다. 웹 메소드로 정의되었지만 현재 웹 서비스 클래스로 상속된 클래스들은 현재 클래스에 대한 웹 서비스에서는 제외된다.

**예제:** 메소드나 클래스 메소드에서, 오브젝트나 데이터 집합의 입력력 인수를 갖지 않는 경우, WebMethod 키워드만을 설정하면 된다:

```
/// This method returns tomorrow's price for the requested stock.  
/// This method is provided free from any warranty or promises.  
ClassMethod Forecast(StockName As %String) As %Integer [WebMethod]  
{  
    // apply patented, nonlinear, heuristic to find new price  
    Set price = $Random(1000)  
    Quit price  
}
```

웹 서비스에서 이 메소드를 호출할 때, 메소드는 다음과 같은 형태의 SOAP 메시지를 웹 서비스로 보낸다:

```
<?xml version="1.0" encoding="UTF-8" ?>  
<SOAP-ENV:Envelope  
    xmlns:SOAP-ENV='http://schemas.xmlsoap.org/soap/envelope/'  
    xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'  
    xmlns:s='http://www.w3.org/2001/XMLSchema'>  
    <SOAP-ENV:Body>  
        <Forecast xmlns='http://tempuri.org'>  
            <StockName xsi:type="s:string">GZP</StockName>  
        </Forecast>  
    </SOAP-ENV:Body>  
</SOAP-ENV:Envelope>
```

웹 서비스는 요청된 작업을 실행하고, 다음과 같은 형태의 회답 SOAP 메시지를 보낸다 :

```
<?xml version="1.0" encoding="UTF-8" ?>

<SOAP-ENV:Envelope
    xmlns:SOAP-ENV='http://schemas.xmlsoap.org/soap/envelope/'
    xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
    xmlns:s='http://www.w3.org/2001/XMLSchema'>
    <SOAP-ENV:Body>
        <ForecastResponse xmlns="http://www.myapp.org">
            <ForecastResult>799</ForecastResult>
        </ForecastResponse>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

## 2) 참조 또는 Output 인수로 값 리턴

참조 또는 Output 타입으로 값을 리턴하는 경우, 웹 메소드의 인수 부분에 Output 또는 ByRef 키워드를 사용한다. 이러한 설정은 SOAP 응답 메시지에 반영된다.

리턴 값을 갖는 다음과 같은 웹 메소드를 예로 보자:

```
Method Dolt() As %String [ WebMethod ]
{
    set ans="Answer"
    Quit ans
}
```

이 웹 메소드는 다음과 같은 응답 메시지를 생성한다:

```
<SOAP-ENV:Body>
    <DoltResponse xmlns="http://www.myapp.org">
        <DoltResult>Answer</DoltResult>
    </DoltResponse>
</SOAP-ENV:Body>
```

반면에, 다음과 같이 리턴 값을 정의하지 않고 Output 타입의 인수로 값을 전달하도록 정의된 웹 메소드의 경우,

```
Method Dolt2(Output ans as %String) [ WebMethod ]
{
    set ans="Answer"
    Quit
}
```

다음과 같은 모양의 응답 메시지가 생성된다:

```
<SOAP-ENV:Body>
  <Dolt2Response xmlns="http://www.myapp.org">
    <ans>Answer</ans>
  </Dolt2Response>
</SOAP-ENV:Body>
```

엘리먼트 `<methodnameResponse>` 의 하위 노드 `<ans>` 은 메소드 Output 의 변수명이다. 반면, 리턴 값으로 리턴하는 메소드가 생성한 메시지에서는 엘리먼트 `<methodnameResponse>` 의 하위 노드가 `<methodnameResult>` 임을 볼 수 있다.

Output 대신 `ByRef` 키워드를 사용하더라도, 같은 형식의 응답 메시지가 생성된다.

단일 웹 메소드에 여러 개의 Output 파라미터를 정의할 수 있으며, 필요한 경우, Input 파라미터도 설정할 수 있다.

### 3) Input/Output 인수값으로 특수문자 사용

XML에서는 (캐리지 리턴, 라인 피드, 탭을 제외한) ASCII 32 이하의 특수 문자들은 허용되지 않는다.

SOAP 메시지에 이러한 특수 문자를 갖는 프로퍼티를 포함하려면, 프로퍼티 타입은 `%Binary` 또는 `%xsd.base64Binary` 이어야 한다. 이 값은 XML로 보내질 때 자동으로 base-64 인코딩으로 변환된다(또한 이 타입의 값을 가져올 때도 자동적으로 base-64 인코딩으로부터 변환된다).

### 4) 오브젝트를 Input/Output 인수로 사용

오브젝트를 웹 메소드의 인수 또는 리턴 값으로 사용하는 경우, 해당 오브젝트는 반드시 XML 구성 가능하여야 한다. 즉, 클래스 정의는 `%XML.Adaptor` 를 상속받아 클래스의 인스턴스가 XML로 전환 가능하도록 해야 한다.

예를 들어, 두 개의 복소수(실수부와 허수부로 구성된 수)를 입력 값으로 하고, 두 수의 합을 리턴하는 웹 메소드를 생각해 보자:

```
ClassMethod Add(a As ComplexNumber,b As ComplexNumber)
As ComplexNumber [WebMethod]
{
  Set sum = ##class(ComplexNumber).%New()
  Set sum.Real = a.Real + b.Real
  Set sum.Imaginary = a.Imaginary + b.Imaginary
  Quit sum
}
```

여기서 `ComplexNumber` 클래스 정의는 다음과 같으며, 이 클래스가 `%XML.Adaptor`를 상속받는 경우에만, 이 메소드를 웹 메소드로 사용할 수 있다.

```

/// A complex number
Class MyApp.ComplexNumber Extends (%RegisteredObject,%XML.Adaptor)
{
    Property Real As %Float;
    Property Imaginary As %Float;
}

```

## 5) 컬렉션을 Input/Output 인수로 사용

컬렉션 타입(%ListOfObjects 또는 %ArrayOfObject)을 인수나 리턴 값으로 사용하여야 하는 경우, 반드시 컬렉션의 *ELEMENTTYPE* 파라미터를 XML 구성 가능 클래스로 설정하여야 한다. 그렇지 않은 경우, WSDL은 필요로 하는 모든 타입들을 정의할 수 없으며 웹 서비스도 사용할 수 없다.

%ListOfObjects 타입에 대하여 두 가지 다른 방법으로 정의할 수 있다:

- 메소드 서명(Signature)에 *ELEMENTTYPE* 파라미터 값을 설정한다.

```

ClassMethod MyMethod() As %ListOfObjects(ELEMENTTYPE="MyApp.MyXMLType")
[ WebMethod ]
{
    //method implementation
}

```

- %ListOfObjects의 하위 클래스를 생성하여 하위 클래스에 파라미터 값을 정의한다. 예를 들어,

```

Class MyApp.MyListOfObjects Extends %ListOfObjects
{
    Parameter ELEMENTTYPE = "MyApp.MyXMLType";
}

```

그리고 이 서브 클래스를 메소드 서명에 리턴 타입으로 사용한다.

```

ClassMethod MyMethod() As MyApp.MyListOfObjects [ WebMethod ]
{
    //method implementation
}

```

%ArrayOfObjects 에 대해서는 하위클래스를 생성하고 사용하여야 한다.

## 6) 데이터 집합을 Input/Output 인수로 사용

이 섹션에서는 웹 메소드에 데이터 집합을 인수 또는 리턴 값으로 사용하는 방법에 대해 다음과 같은 항목별로 설명한다 :

- %XML.DataSet 소개
- 정형화된 데이터 집합 정의

- 데이터 집합을 리턴하는 메소드 생성
- 데이터 집합을 Input 인수로 사용하는 메소드 생성
- 데이터 집합 이용
- 데이터 집합 타입을 제어하는 방법
- 데이터 집합을 XML로 보는 유틸리티 메소드

## %XML.DataSet 소개

일반적으로, 웹 메소드의 인수나 리턴 값으로 사용하려고 하는 데이터 집합을 생성하려면, %XML.DataSet 또는 사용자 정의 하위 클래스를 사용한다. 이 클래스는 표준 %XML.DataSet 의 XML 구성 가능 하위 클래스로서, %ResultSet를 사용하는 방법과 거의 유사하게 사용하면 된다.

**참고:** %XML.DataSet 의 WSDL 스키마 및 XML 형식은 Microsoft 의 데이터 집합 형식으로서, Caché 나 .NET 웹 클라이언트에서만 이 형식을 이해할 수 있다. 쿼리의 결과를 자바 기반의 웹 클라이언트에서 사용할 수 있도록 하기 위하여는 %ListOfObjects 하위 클래스를 사용한다 – SAMPLES 네임스페이스의 SOAP.Demo 에서 예를 찾을 수 있다.

## 정형화된 데이터 집합 정의

모든 데이터 집합은 데이터 검색을 위한 쿼리를 사용하는데, 이 쿼리가 컴파일 시점에 정의되어 있으면 데이터 집합은 정형화되었다고 하고, 그렇지 않은 경우 비정형 데이터 집합이라 한다. 정형화된 데이터 집합은 많은 경우에 편리한데, 예를 들면 .NET에서는 Microsoft Visual Studio에서 코드 완료(?)를 이룰 수 있다.

정형화된 데이터 집합을 정의하기 위해서는, %XML.DataSet 의 하위 클래스를 생성하고 *QUERYNAME* 및 *CLASSNAME* 파라미터에 설정하는데 이 값들은 특정 SQL 쿼리를 참조한다.

**참고:** %XML.DataSet를 메소드의 리턴 값으로 사용하는 경우, 그 값의 XML 타입은 DataSet이 된다. %XML.DataSet 의 하위 클래스를 리턴 값으로 사용하는 경우에는, 그 값의 XML 타입은 그 하위 클래스의 이름이 된다. 이러한 설정은 다른 XML 구성 가능 클래스들에 있어서도 동일하며 WSDL에 사용되는 XML 타입에도 적용된다.

## 데이터 집합 리턴 메소드 정의

데이터 집합을 리턴하는 웹 메소드는 다음과 같이 정의 한다:

1. 리턴 타입은 %XML.DataSet 이거나 사용자 정의 하위 클래스이어야 한다.
2. 웹 메소드는 반드시 %XML.DataSet 또는 사용자 정의 하위 클래스의 인스턴스를 생성하고, 이 인스턴스를 쿼리를 실행하는데 필요한 정보로 초기화하여야 한다. 다음의 옵션은 %ResultSet 클래스와 동일한 옵션이다:

- 특정 클래스내에서 클래스 쿼리를 정의하였다면, 그 클래스 쿼리는 다음과 같이 사용한다. **%New()** 메소드를 통해 데이터 집합을 생성하고, 생성된 데이터 집합의 *ClassName*과 *QueryName* 프로퍼티를 설정한다. 여기에서 클래스 쿼리에 설정된 쿼리명은 해당 클래스에서 정의한 클래스 쿼리를 말한다:

```
// Get all saved people; technique 1 for the doc
Method GetPeople1() As %XML.DataSet [ WebMethod ]
{
    set dataset=##class(%XML.DataSet).%New()
    set dataset.ClassName="Sample.Person"
    set dataset.QueryName="GetPeople"
    Quit dataset
}
```

이 메소드의 다양한 활용은 뒤에 소개되는 “예제 2: 정형화된 데이터 집합”을 참조하기 바란다.

- 위의 코드를 다음과 같이 클래스명 및 쿼리명을 직접 **%New()** 메소드의 인수로 설정하고 데이터 집합을 생성할 수 있다:

```
// Get all saved people; technique 2 for the doc
Method GetPeople2() As %XML.DataSet [ WebMethod ]
{
    set dataset=##class(%XML.DataSet).%New("Sample.Person:GetPeople")
    Quit dataset
}
```

- %New()** 메소드의 인수로 "%DynamicQuery:SQL" 를 주고 데이터 집합을 생성할 수도 있는데, 이 경우 생성된 데이터 집합의 **Prepare()** 메소드를 호출하여 직접 SQL 쿼리문을 설정한다:

```
// Get all saved people; technique 3 for the doc
Method GetPeople3() As %XML.DataSet [ WebMethod ]
{
    set dataset=##class(%XML.DataSet).%New("%DynamicQuery:SQL")
    set mysql = "SELECT Name,DOB FROM Sample.Person"
    set status=dataset.Prepare(mysql)
    if $$$ISERR(status) do ..StatusError(status)
    Quit dataset
}
```

**Prepare()** 메소드는 주어진 쿼리문을 검토하고 그 결과(status)를 리턴하는데, 위의 예와 같이, **\$\$\$ISERR()** 메소드를 통해 리턴값을 확인하고, 오류인 경우 웹 서비스의 정의된 메소드 **StatusError()** 를 이용하여 오류 메시지를 얻는다.

- 생성된 데이터 집합의 **Execute()** 메소드를 호출하여 주어진 쿼리를 실행한다. 이 때, 주어진 쿼리에 제공되어져야 하는 인수들이 있는 경우, 이 값을 메소드의 인수로 정의된 순서대로 제공한다. 메소드는 쿼리 실행의 성공 여부를 리턴하며, 이 값은 위의 설명대로 반드시

확인되어져야 한다.

4. 메소드를 종료하면서 데이터 집합을 리턴한다.

#### 예제 1: 비정형(untyped) 데이터 집합

다음은 비정형(untyped) 데이터 집합을 반환하는 예이다:

```
// Get all people who have a given home ZIP code
Method GetPeopleByZip(zip As %String) As %XML.DataSet [ WebMethod ]
{
    Set dataset=##class(%XML.DataSet).%New("%DynamicQuery:SQL")
    Set sql1 = "SELECT Name FROM Sample.PersonWithAddress"
    Set sql2 = " where Sample.PersonWithAddress.Home_Zip = "_zip
    Set status=dataset.Prepare(sql1_sql2)
    If $$$$ISERR(status) do ..StatusError(status)
    Set status=dataset.Execute()
    If $$$$ISERR(status) do ..StatusError(status)
    Quit dataset
}
```

앞에서 설명한 바와 같이, StatusError() 는 오류 메시지를 추출하여 SOAP Fault로 리턴하는 메소드이다.

#### 예제 2: 정형화된(typed) 데이터 집합(셋)

먼저 %XML.DataSet 의 하위 클래스를 정의하고 쿼리를 설정한다:

```
Class Sample.PeopleDataSet Extends %XML.DataSet
{
    Parameter QUERYNAME = "GetPeople";
    Parameter CLASSNAME = "Sample.Person";
}
```

여기에서, Sample.Person 클래스가 클래스 쿼리 GetPeople을 가지고 있다고 가정한다.

다음의 웹 메소드는 앞의 하위 클래스를 사용하여 정형화된 데이터 집합을 리턴한다:

```
// Get all saved people; technique 4 for the doc
Method GetPeople4() As Sample.PeopleDataSet [ WebMethod ]
{
    set dataset=##class(Sample.PeopleDataSet).%New()
    Quit dataset
}
```

## 데이터 집합을 Input 인수로 사용하는 메소드

데이터 집합을 웹 메소드의 input 인수로 사용하려면, 해당 데이터 집합은 %XML.DataSet 을 상속받아야 한다. 예를 들면, 다음과 같이 웹 메소드의 서명을 정의한다:

```
ClassMethod ProcessDataSet(dataset as %XML.DataSet) As %String [ WebMethod ]
{
}
```

다음 섹션에서는 메소드에서 데이터 집합을 사용할 수 있는 방법에 대해 설명한다.

### 데이터 셋의 이용

%XML.DataSet 클래스는 데이터 집합 제어를 위해 %ResultSet 클래스와 동일한 인터페이스를 제공한다. 특히 한 번에 한 행씩 데이터 집합을 읽어가는데 주어진 컬럼에서의 값과 같은 정보를 검색할 수 있는 메소드를 사용한다. 다음 행을 읽으려면, **Next()** 메소드를 사용하는데, 일반적으로 데이터 집합의 마지막에 도달할 때까지(**Next()** 가 0을 반환하는 시점) 각 행을 반복적으로 계속 읽어 나간다.

중간에 오류가 발생하는 경우에도, **Next()** 메소드는 0을 반환한다.

메소드가 완료되면 데이터 집합 인스턴스는 자동 소멸 되는데, 데이터 집합을 메소드 내에서 제거하려면, **Close()** 메소드를 사용한다.

#### ➤ 일반적(대표적인) 구조

다음은 데이터 집합을 사용하는 메소드의 일반적인 개요를 보여주는 예이다:

```
ClassMethod ProcessDataSet(dataset as %XML.DataSet) As %String [ WebMethod ]
{
    Set returnvalue=""
    While (dataset.Next(.status))
    {
        //do stuff and set returnvalue
    }
    Quit returnvalue
}
```

#### ➤ 예제

다음은 주어진 데이터 집합의 모든 행을 카운트하는 간단한 예이다:

```
/// count rows in dataset
ClassMethod CountRows(dataset as %XML.DataSet) As %Numeric [ WebMethod ]
{
    Set count=0
    While (dataset.Next(.status))
    {
        Set count=count+1
    }
    Quit count
}
```

입력값이 오브젝트이기 때문에 이 메소드를 테스트하기 위해 생성된 테스트 웹 페이지를 사용할 수 없다.

#### ➤ 데이터 셋 검사에 사용되는 메소드

데이터 집합은 데이터 집합의 데이터를 검색하는데 사용되는 공통의 메소드들을 상속받는다. 다음은 가장 많이 사용되는 메소드들이다:

- **GetColumnCount()**: 데이터 집합의 컬럼수를 반환한다.
- **Get()**: 현재 행에서 주어진 컬럼명의 컬럼 데이터를 반환한다.
- **GetData()**: 현재 행에서 주어진 컬럼번호의 컬럼 데이터를 반환한다. 첫번째 컬럼의 컬럼번호는 1이다.
- **GetColumnName()**: 주어진 컬럼번호의 컬럼 헤더를 반환한다.
- **GetColumnHeader()**: 주어진 컬럼번호의 컬럼 명을 반환한다.
- **GetColumnType()**: 주어진 컬럼번호의 컬럼 데이터 타입을 반환하는데 이 리턴값은 특정 데이터 타입에 대응하는 정수값이다. 각 데이터 타입 및 대응하는 정수값에 대한 정보는 %Library.ResultSet 클래스 문서를 참조하기 바란다.

#### 데이터 셋 형식의 제어

기본적으로 데이터 집합은 Microsoft DiffGram 형식으로 정의되었으며, XML 스키마로 시작한다. 다음은 그 예이다:

```
<SOAP-ENV:Body>
<Get0Response xmlns="http://www.myapp.org">
  <Get0Result>
    <s:schema id="DefaultDataSet" xmlns="">
      attributeFormDefault="qualified"
      elementFormDefault="qualified"
      xmlns:s="http://www.w3.org/2001/XMLSchema"
      xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
        <s:element name="DefaultDataSet" msdata:IsDataSet="true">
          <s:complexType>
            <s:choice maxOccurs="unbounded">
              <s:element name="GetPeople">
                <s:complexType>
                  <s:sequence>
                    <s:element name="Name" type="s:string" minOccurs="0" />
                    <s:element name="DOB" type="s:date" minOccurs="0" />
                  </s:sequence>
                </s:complexType>
              </s:element>
            </s:choice>
          </s:complexType>
        </s:element>
      </s:schema>
```

```

<diffgr:diffgram
  xmlns:msdata="urn:schemas-microsoft-com:xml-msdata"
  xmlns:diffgr="urn:schemas-microsoft-com:xml-diffgram-v1">
  <DefaultDataSet xmlns="">
    <GetPeople diffgr:id="GetPeople1" msdata:rowOrder="0">
      <Name>Quine,Howard Z.</Name>
      <DOB>1965-11-29</DOB>
    </GetPeople>
    ...
  </DefaultDataSet>
</diffgr:diffgram>
</Get0Result>
</Get0Response>
</SOAP-ENV:Body>

```

%XML.DataSet 클래스에는 이러한 형식을 제어하기 위해 다음과 같은 옵션들을 제공한다:

- *DATAONLY* 파라미터와 *DiffGram* 프로퍼티로 DiffGram 형식으로 출력할지 여부를 제어한다.. %XML.DataSet의 하위 클래스를 생성하고 *DATAONLY*를 1로 설정하거나, *DiffGram*을 0으로 설정하면, Output은 DiffGram 형식이 되지 않는다. 대신 XML 데이터 집합은 다음과 같다:

```

<SOAP-ENV:Body>
  <Get0Response xmlns="http://www.myapp.org">
    <Get0Result>
      <GetPeople xmlns="">
        <Name>Quine,Howard Z.</Name>
        <DOB>1965-11-29</DOB>
      </GetPeople>
      <GetPeople xmlns="">
      ...
    </Get0Result>
  </Get0Response>
</SOAP-ENV:Body>

```

DiffGram 형식과는 대조적으로, 스키마는 기본적으로 출력되지 않으며, 출력시에도 <diffgram> 엘리먼트를 포함하지 않는다.

- *NeedSchema* 프로퍼티는 Output이 XML 스키마를 포함할지 여부를 결정한다. DiffGram 형식을 사용한다면, 디폴트로 스키마는 출력되며, DiffGram 형식을 사용하지 않는 경우, 스키마는 출력되지 않는다. 스키마 출력을 강제하려면, *NeedSchema*를 1로 설정하며, 출력하지 않으려면 0으로 설정하면 된다.
- DiffGram 형식을 사용하는 경우, *WriteEmptyDiffgram* 프로퍼티는 데이터 집합이 행을 갖지 않을 때 <diffgram> 엘리먼트의 내용을 제어한다. 기본적으로(또는 *WriteEmptyDiffgram=0*인 경우) <diffgram> 엘리먼트는 다음과 같은 빈 엘리먼트를 가지게 된다:

```
...
<diffgr:diffgram xmlns:msdata="urn:schemas-microsoft-com:xml-msdata"
    xmlns:diffgr="urn:schemas-microsoft-com:xml-diffgram-v1">
    <DefaultDataSet xmlns="">
    </DefaultDataSet>
</diffgr:diffgram>
...
```

반면에 WriteEmptyDiffgram=1 인 경우, <diffgram> 엘리먼트는 다음과 같이 아무것도 갖지 않는다:

```
...
<diffgr:diffgram xmlns:msdata="urn:schemas-microsoft-com:xml-msdata"
</diffgr:diffgram>
...
```

DiffGram 형식을 사용하지 않는 경우에는, 이 프로퍼티는 아무 역할도 하지 않는다.

- DiffGram 형식을 사용하는 경우, *DataSetName* 프로퍼티는 <diffgram> 엘리먼트내의 엘리먼트의 이름을 제어한다. 위의 예와 같이 이 요소의 디폴트 이름은 <DefaultDataSet>이며 DiffGram 형식을 사용하지 않는 경우, 이 프로퍼티는 아무 역할도 하지 않는다.

## 데이터 셋과 스키마를 XML로 보기

%XML.DataSet을 확장한 데이터 집합에는 XML 생성에 사용되는 유ти리티 메소드들이 포함되어 있다. 이 메소드들은 모두 현재의 디바이스로 출력한다:

- **WriteXML()** 데이터 집합을 XML로 쓰는데, 선택적으로 XML 스키마가 먼저 올 수 있다. 이 메소드에는 상위-레벨 태그 이름, 네임스페이스 사용, 널 처리등을 제어하는 인수들이 있다. 기본적으로, 이 메소드는, 앞에서 설명된 데이터 집합 형식에 따라 실행된다. 그렇지만 출력을 DiffGram 형식으로 할 것인지를 비롯한 출력을 제어하는 인수들을 사용하여 메소드 실행시 출력 형식을 변경할 수도 있다.
- **XMLExport()** 데이터 집합에 대한 XML 스키마를 출력한 후 XML 형식의 데이터 집합을 내보낸다.
- **WriteSchema()** 데이터 집합에 대한 XML 스키마만을 출력한다.
- **XMLSchema()** 데이터 집합 클래스에 대한 Microsoft 기준 XML 표현으로 출력한다.

## 7) 클래스 쿼리를 웹 메소드로 사용하기

정형화된 데이터 집합을 반환하는 클래스 쿼리를 웹 메소드로 사용할 수 있다. 다음과 같이, 간단히 WebMethod 키워드를 쿼리 정의에 추가하면 된다:

```
Query QueryByName(name As %String = "") As  
%SQLQuery(CONTAINID = 1, SELECTMODE = "RUNTIME") [WebMethod]  
{  
    SELECT ID, Name, DOB, SSN  
    FROM Sample.Person  
    WHERE (Name %STARTSWITH :name)  
    ORDER BY Name  
}
```

이 웹 메소드가 실행되면, 다음과 같이 SOAP 형식의 데이터 집합이 반환한다:

```
<SOAP-ENV:Envelope  
xmlns:SOAP-ENV='http://schemas.xmlsoap.org/soap/envelope/'  
xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'  
xmlns:s='http://www.w3.org/2001/XMLSchema'>  
  
<SOAP-ENV:Body>  
<QueryByNameResponse xmlns="http://www.myapp.org"><QueryByNameResult>  
<s:schema id="QueryByName_DataSet"  
targetNamespace="http://www.myapp.org/QueryByName_DataSet"  
xmlns="http://www.myapp.org/QueryByName_DataSet"  
xmlns:mstns="http://www.myapp.org/QueryByName_DataSet"  
attributeFormDefault="qualified" elementFormDefault="qualified"  
xmlns:s="http://www.w3.org/2001/XMLSchema"  
xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">  
    <s:element name="QueryByName_DataSet" msdata:IsDataSet="true">  
        <s:complexType>  
            <s:choice maxOccurs="unbounded">  
                <s:element name="QueryByName">  
                    <s:complexType>  
                        <s:sequence>  
                            <s:element name="ID" type="s:long" minOccurs="0" />  
                            <s:element name="Name" type="s:string" minOccurs="0" />  
                            <s:element name="DOB" type="s:date" minOccurs="0" />  
                            <s:element name="SSN" type="s:string" minOccurs="0" />  
                        </s:sequence>  
                    </s:complexType>  
                </s:element>  
            </s:choice>  
        </s:complexType>  
        <s:unique name="Constraint1" msdata:PrimaryKey="true">  
            <s:selector xpath=".//mstns:QueryByName" />  
            <s:field xpath="mstns:ID" />  
        </s:unique>  
    </s:element>  
</s:schema>  
<diffgr:diffgram xmlns:msdata="urn:schemas-microsoft-com:xml-msdata"  
xmlns:diffgr="urn:schemas-microsoft-com:xml-diffgram-v1">
```

```

<QueryByName_DataSet xmlns="http://www.myapp.org/QueryByName_DataSet">
  <QueryByName diffgr:id="QueryByName1" msdata:rowOrder="0">
    <ID>1</ID>
    <Name>Xerxes,Dave L.</Name>
    <DOB>1941-12-09</DOB>
    <SSN>499-23-9251</SSN>
  </QueryByName>
</QueryByName_DataSet>
</diffgr:diffgram>
</QueryByNameResult></QueryByNameResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

## 8) SOAP 메시지에 대한 바인딩 스타일 명사하기

`SoapBindingStyle` 키워드는 웹 메소드의 입출력에 대한 바인딩 스타일을 제어한다. 즉, WSDL 바인딩을 SOAP 메시지로 전환하는 방법을 정의하고, SOAP 메시지 본문 형식을 제어한다. 이 키워드에 대한 값으로 `document` (디폴트 값) 또는 `rpc` 를 설정할 수 있는데, 클래스 레벨 및 메소드 레벨에서 사용 가능하다.

**참고:** 기본적으로 헤더 엘리먼트는 (SOAP-encoded 보다) 리터럴(literal) 포맷으로 되어 있어, `SoapBindingStyle`과는 독립적이다.

### DOCUMENT 스타일의 바인딩

다음의 SOAP 메시지는 디폴트값 `document`로 설정된 `SoapBindingStyle`를 갖는 웹 메소드에 대한 응답으로 생성된 것이다.

```

<?xml version="1.0" encoding="UTF-8" ?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <xns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <xns:s="http://www.w3.org/2001/XMLSchema">
    <SOAP-ENV:Body>
      <DivideResponse xmlns="http://www.myapp.org">
        <DivideResult>.5</DivideResult>
      </DivideResponse>
    </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>

```

### RPC 스타일 바인딩

반면에 다음의 SOAP 메시지는 `rpc`로 설정된 `SoapBindingStyle`를 갖는 웹 메소드에 대한 응답으로 생성된 것이다:

```

<?xml version="1.0" encoding="UTF-8" ?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV='http://schemas.xmlsoap.org/soap/envelope/'
xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xmlns:s='http://www.w3.org/2001/XMLSchema'
xmlns:tns='http://www.myapp.org'>
  <SOAP-ENV:Body>
    <tns:DivideResponse>
      <tns:DivideResult>.5</tns:DivideResult>
    </tns:DivideResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

## 9) SOAP 메시지 인코딩 설정

`SoapBodyUse` 키워드는 웹 메소드 입출력에 대한 인코딩을 제어한다. 이 키워드에 대한 값으로 `literal` (디폴트 값) 또는 `encoded` 를 설정할 수 있으며 클래스 레벨과 메소드 레벨에서 사용 가능하다. .NET를 포함한 대부분의 SOAP 클라이언트는 `literal` 형식을 기본 값으로 가지며 `encoded` 형식은 최신 형식으로 간주되지 않는다.

**참고:** `SoapBodyUse` 키워드는 WSDL에 영향을 미치는 `ELEMENTQUALIFIED` 와 `XMLEMENT` 파라미터의 기본 값에 영향을 준다.

이후의 내용들은 각각의 `SoapBodyUse` 값에 대한 웹 메소드의 다른 출력 형태들의 예에 대한 설명이다. 두 예에서 프로퍼티 `Name`, `DOB`, `FavoriteColors` (문자열 리스트 타입), 및 `Home(Address` 오브젝트) 과 `Office(Address` 오브젝트)를 갖는 동일한 오브젝트 `PersonWithAddress`에 대한 다른 출력 내용을 보여준다.

### `literal` 데이터

다음의 SOAP 메시지는 기본 설정으로 `literal` 값의 `SoapBindingStyle`을 갖는 웹 메소드에 대한 응답으로 생성된 것이다.

```

<?xml version="1.0" encoding="UTF-8" ?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV='http://schemas.xmlsoap.org/soap/envelope/'
xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xmlns:s='http://www.w3.org/2001/XMLSchema'>
  <SOAP-ENV:Body>
    <GetPerson2Response xmlns='http://www.myapp.org'>
      <GetPerson2Result>
        <Name>Wijnschenk, Bob F.</Name>
        <DOB>1963-05-01</DOB>
        <FavoriteColors>
          <FavoriteColorsItem>Orange</FavoriteColorsItem>
        </FavoriteColors>
        <Home>
          <Street>455 Main Blvd</Street>
          <City>Xavier</City>
        </Home>
      </GetPerson2Result>
    </GetPerson2Response>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

```

<State>MD</State>
<Zip>21692</Zip>
</Home>
<Office>
<Street>8074 Madison Place</Street>
<City>Reston</City>
<State>IA</State>
<Zip>29289</Zip>
</Office>
</GetPerson2Result>
</GetPerson2Response>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

### SOAP encoded 데이터

반면에, 다음의 SOAP 메시지는 SoapBodyUse가 **encoded** 값을 가지는 웹 메소드에 대한 응답으로 생성된 것이다.

**참고:** SOAP 1.2에서는 인코딩된 버전이 약간 다름으로 주의해야 한다.

```

<?xml version="1.0" encoding="UTF-8" ?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV='http://schemas.xmlsoap.org/soap/envelope/'
xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xmlns:s='http://www.w3.org/2001/XMLSchema'
xmlns:SOAP-ENC='http://schemas.xmlsoap.org/soap/encoding/'
xmlns:types='http://www.myapp.org/encodedTypes'>
<SOAP-ENV:Body
SOAP-ENV:encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'>
<types:GetPerson2EncResponse>
<GetPerson2EncResult href="#id1" />
</types:GetPerson2EncResponse>
<types:PersonWithAddress id="id1" xsi:type="types:PersonWithAddress">
<Name>Wijnschenk, Bob F.</Name>
<DOB>1963-05-01</DOB>
<FavoriteColors SOAP-ENC:arrayType="s:string[1]">
<FavoriteColorsItem>Orange</FavoriteColorsItem>
</FavoriteColors>
<Home xsi:type="types:Address">
<Street>455 Main Blvd</Street>
<City>Xavier</City>
<State>MD</State>
<Zip>21692</Zip>
</Home>
<Office xsi:type="types:Address">
<Street>8074 Madison Place</Street>
<City>Reston</City>
<State>IA</State>
<Zip>29289</Zip>
</Office>
</types:PersonWithAddress>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

## 4. 웹 클라이언트 생성

웹 클라이언트는 웹 서비스를 액세스하는 소프트웨어로서 각각의 웹 서비스 메소드에 대응하는 프럭시 메소드들을 제공한다. 프럭시 메소드는 웹 서비스 메소드와 동일한 서명을 사용하며, 요청시 웹 서비스 메소드를 호출한다. 이 장에서는 웹 클라이언트의 생성 및 사용에 대해 설명한다.

### 1) Caché SOAP 클라이언트 마법사 개요

Cache 웹 클라이언트를 생성하려면, Caché Studio에서 SOAP 클라이언트 마법사를 사용한다. 마법사는 주어진 WSDL 문서를 사용하여 웹 서비스에 맞는 웹 클라이언트뿐만 아니라 웹 클라이언트 클래스 및 모든 관련 클래스들도 생성한다.

대상 WSDL 문서는 URL 또는 로컬 WDL 파일 경로로 주어진다.

### 2) SOAP 클라이언트 마법사 사용하기

SOAP 클라이언트 마법사를 사용하여 웹 서비스에 대한 클라이언트를 생성하려면

1. Studio에서 작업하고자 하는 Caché 네임스페이스에 있는지 확인 한다.
2. 메뉴 도구 > Add-ins > SOAP Client Wizard 를 선택한다.
3. 첫 화면에서 사용하려고 하는 웹 서비스에 대한 WSDL 문서의 URL을 입력한다. WSDL 문서가 사용자명 및 패스워드로 보안 설정이 되어있으면, 해당 사용자명 및 패스워드를 URL 끝에 첨부한다.

예를 들면, 웹 서비스가 보안 설정된 Caché 시스템 내에 있는 경우, 다음과 같이 Caché 사용자명 및 패스워드를 첨부한다:

```
&CachéUserName=_SYSTEM&CachéPassword=SYS
```

또는, CSP 응용프로그램에 대해서는(에서) 패스워드 보안 설정을(장치를) 하지 않을 수도 있(는)다.

모든 경우에 사용할 수 있는 또 다른 옵션으로, WSDL 문서를 파일로 저장하여 해당 파일을 읽어드리는 것이다. 이 경우, SOAP 클라이언트 마법사 대신 클래스 **%SOAP.WSDL.Reader** 의 클래스 메소드 **Process()** 를 사용할 수도 있다.

4. ‘다음’을 클릭한다.

WSDL이 SSL을 사용하는 장소에 있다면 (즉, URL이 <https://>로 시작한다면), 마법사는 SSL 설정 화면을 먼저 보여줄 것이다.

5. 현재의 Secure Socket Layer(SSL)이나 Transport Layer Security(TLS)의 구성명을 입력하여 연결에 대한 인증을 받는다.

SSL/TLS 생성 및 관리에 대한 정보는 온라인 문서 Caché Security Administration Guide의 Using SSL/TLS with Caché 부분을 참조하기 바란다.

6. ‘다음’을 클릭한다.
7. **클래스 타입**, 웹 클라이언트 클래스에 대한 타입을 선택한다: registered (디폴트), persistent, serial.
8. 프럭시 클래스 패키지, 웹 클라이언트의 패키지명으로서 생성된 타입 클래스에 대한 기본 패키지명으로도 사용된다. 디폴트 패키지명은 서비스명이다.  
이 패키지명이 기존 패키지명과 동일한 경우, 마법사는 기존 클래스를 덮어쓴다.
9. ‘다음’을 클릭한다. 마법사는 클래스를 생성, 컴파일하며 생성된 클래스 목록을 보여준다.
10. ‘마침’을 클릭한다.

**참고:** 웹 서비스가 보안 설정된 Caché 내에 존재할 때, URL에 사용자명과 패스워드가 주어지지 않으면, 마법사는 다음과 같은 오류 메시지를 발생시킨다:

```
ERROR #6301: SAX XML Parser Error: Expected entity name for reference  
while processing Anonymous Stream at line 10 offset 27
```

# 제 9 장      Zen 어플리케이션 개발

## 1. Zen 소개

- 1) Zen 데모
- 2) 지원 브라우저
- 3) Zen 커뮤니티
- 4) Zen의 장점
- 5) 참고 문헌

## 2. Zen 컴포넌트 이용하여 개발하기

- 1) Zen 테이블
- 2) Zen report 만들기

## 1. Zen 소개

Zen 응용프로그램 프레임워크은 사전 정의된 오브젝트 컴포넌트들을 결합하여, 복합적이며 데이터가 풍부한 웹 어플리케이션을 신속하고 간단하게 개발할 수 있게 해주는 개발환경이다. 제공되는 컴포넌트들은 복합 웹 어플리케이션에 필요한 표준 HTML과 자바스크립트를 자동적으로 생성한다. 또한, 사용자 브라우저와 서버에서 실행되는 응용프로그램 로직 사이에서 공유되는 공통 오브젝트 모델을 제공한다.

Zen은 인터시스템즈의 CSP(Caché Server Page) 와 Caché 오브젝트 데이터베이스 기술을 기반으로 하고 있다. 이 기술들은 사용자로 하여금 웹 어플리케이션 수행에 필요한 안정적이고 확장성 및 이식성이 뛰어난 플랫폼을 제공한다. Zen은 CSP를 대체하는 것이 아니라 CSP가 제공하는 기본적인 기능 - 성능, 데이터 액세스, 보안, 협지화, 환경 구성 등 - 들을 사용하여 제공되는 보다 용이한 웹 기반 응용프로그램 개발 환경이다.

### 1) Zen 데모

Zen에는 다음과 같이 테스트해 볼 수 있는 샘플 응용프로그램을 제공한다:

1. Firefox나 Internet Explore를 사용, 브라우저를 시작한다.
2. 다음의 URI를 입력한다.

<http://localhost:57772/csp/samples/ZENDemo.Home.cls>

여기에서 57772는 사용자가 지정한 Caché 웹 서버 포트 번호이다.

설정된 포트 번호는 시스템 관리 포탈 홈 페이지 상단의 ‘정보’를 선택하면 정의된 웹 서버 포트값을 확인할 수 있다.

### 2) 지원 브라우저

지원 브라우저 목록은 온라인 문서 InterSystems Supported Platforms에서 확인하기 바란다.

**참고:** InterSystems는, Zen 어플리케이션이 여러 범용 브라우저에서 모든 실행가능하도록 업계 표준 – XML, HTML, JavaScript, CSS(Cascading Style Sheets) 등 – 을 기반으로 하고 있다. 브라우저가 이 표준들에 대한 다양한 다른 레벨을 지원하기 때문에, 이들간의 레이아웃 차이는 피할 수 없는데, 이러한 차이를 피하기 위해서는 개발 단계에서 가급적 다양한 브라우저에서 테스트해 보기 권장한다.

### 3) Zen 커뮤니티

Zen 커뮤니티는 Zen 사용자들이 다양한 정보 및 질문과 답을 얻고, 코드와 경험을 공유하는 온라인 포럼으로서 InterSystems 고객과 직원이면 누구나 참여할 수 있는 열린 공간이다. Zen 커뮤니티는 재사용 가능한 Zen 컴포넌트 및 코드 예제, 그리고 멤버들간의 다양한 질의 응답 자료들을 제공한다.

Zen 커뮤니티는 구글 그룹이기 때문에 참여하기 위해서는 반드시 구글 그룹 계정을 만들어야 한다.

Zen 커뮤니티로의 액세스는 다음 URI를 사용한다:

<http://www.intersystems.com/community/zen>

#### 4) Zen의 장점

- **빠른 개발:** 사전 정의된 컴포넌트들을 조합하는 형태의 개발은 사용자 인터페이스를 신속하게 생성할 수 있는 검증된 방법으로서 Zen은 웹 기반의 응용프로그램 개발에 이 방법을 제공한다.
- **단순성:** Zen은 표준 HTML 클라이언트를 사용함으로써, 별도의 클라이언트 컴포넌트를 필요로 하지 않는다. 또한 서버와 클라이언트 사이에 중간 계층이 없으므로 개발, 배치, 지원이 보다 용이하다.
- **확장가능 컴포넌트 라이브러리:** Zen은 사전 정의된 풍부한 컴포넌트 라이브러리를 제공하는데 이에는 모든 표준 콘트롤 타입 뿐만 아니라, 데이터 인식 콤보 상자, 테이블, 그리드, 템, 트리 콘트롤, 메뉴, 그룹 컴포넌트등이 포함되어 있다.
- **오브젝트 데이터베이스 통합:** Zen은 Caché 오브젝트 데이터베이스 기반의 개발 환경이기 때문에 Caché의 데이터 오브젝트를 통해 Zen 클라이언트와 서버는 커뮤니케이션을 진행한다. 암호화, 데이터 관리등과 같은 기본적인 세부 사항은 검증된 Caché 기술을 사용하여 자동적으로 처리된다.
- **코드 생성:** 대부분의 코드와 비즈니스 로직은 일관성과 신속한 개발을 보장하는 상위 레벨 모델링으로부터 자동적으로 생성된다.
- **확장성:** 표준 CSS 기능을 사용하여, Zen 응용프로그램의 모양과 느낌을 쉽게 수정할 수 있다 – Zen 컴포넌트에 다른 속성을 주거나, 사용자 정의 컴포넌트의 개발.
- **SVG의 통합 사용:** Zen은 SVG (Scalable Vector Graphics) 컴포넌트들을 사용하여 웹 어플리케이션에서의 대화형의 그래픽을 구현하는데 이러한 컴포넌트로는 사전 정의된 차트, 도수계등이 있으며 새로운 그래픽 컴포넌트를 개발할 수도 있다.
- **클라이언트 독립성:** Zen은 Firefox 나 Internet Explorer 등 범용 브라우저들 간의 차이점에 영향받지 않는 컴포넌트들을 제공함으로써 웹 어플리케이션이 특정 브라우저의 속성에 의존하지 않고 실행할 수 있게 하여준다.
- **보안:** Zen은 Caché 오브젝트 데이터베이스의 보안 모델과 긴밀하게 통합되어 있다.
- **폼 사용:** Zen 프레임워크는 데이터 출력 및 편집을 제어할 수 있는 다양한 콘트롤들을 갖는 폼을 정의할 수 있도록 한다. 프레임워크는 폼으로의 데이터 로딩, 폼 데이터의 검증 및 저장을 위한 다양한 지원을 제공한다..
- **페이지 레이아웃:** Zen에는 웹 페이지 컴포넌트의 분류와 배치를 정의하는 확장 프레임워크를 제공한다.
- **이벤트 관리:** Zen 응용프로그램은 사용자 이벤트에 대한 컴포넌트 대응 방법을 쉽게 정의할 수 있는데, 모든 이벤트는 클라이언트 브라우저 또는 데이터 서버상에서 실행되도록 정의할 수 있는 메소드의 호출로 처리된다.
- **다국어 지원:** 다국어를 지원해야 하는 어플리케이션을 위해 Zen은 현지어로 번역되어져야 하는 제목이나 캡션을 체크하여 해당 데이터의 데이터베이스를 자동적으로 생성하는 메커니즘을 갖고 있다. 이 데이터베이스는 XML로 출력 가능하며, 다른 언어로 번역할 수도 있다. 실행시 Zen은 사용자가 원하는 언어로 페이지를 자동 표시한다.
- **문서 출력:** Zen은 데이터를 XHTML 또는 PDF 형식의 보고서로 출력할 수 있는 확장 프레임워크를 제공한다.

## 5) 참고 문헌

Zen 개발을 위해서는 다음 주제들에 대한 충분히 숙지가 필요하다:

- HTML(HyperText Markup Language), XML(eXtensible Markup Language), JavaScript, CSS(Cascading Style Sheets)- 인터넷 및 서점에서 관련 책자들을 쉽게 얻을 수 있다.
- Caché, Caché ObjectScript, Caché Server Pages, and Caché SQL. 사용자 경험 정도에 따라, 인터시스템즈의 온라인 문서들중 다음 문서들을 활용한다:
  - ✓ Using Caché Objects
  - ✓ Using Caché ObjectScript
  - ✓ Using Caché Server Pages (CSP)
  - ✓ Using Caché SQL

## 2. Zen 컴포넌트 이용하여 개발하기

이 장에서는 여러 Zen 컴포넌트 중, 한국 웹 환경에서 가장 많이 사용될 수 있는 Zen 테이블에 관해 자세히 살펴보기로 한다. 다른 컴포넌트들에 대한 정보는 Caché 온라인 문서 Using Zen Components 를 참조하기 바란다.

### 1) Zen 테이블

Zen 테이블은 데이터 주도형으로, SQL 쿼리에 의해 반환되는 ResultSet을 HTML 테이블 형식으로 출력한다. Zen 테이블에서 ResultSet을 생성하는 방법에는 여러가지가 있는데, 예를 들어, SQL 문을 명시하거나, 사전 정의된 SQL 쿼리 참조하는 방법, 또는 SQL 쿼리 생성에 필요한 속성들을 Zen 입력값으로 제공하는 방법등이다.

일단 데이터를 얻게되면, 사용자가 원하는 방법으로 결과 테이블의 스타일을 정의할 수 있는데, 다음은 이렇게 생성된 간단한 예이다. 이 테이블에서는 행 교체에 대해 얼룩 무늬 패턴을 사용하였으며, 이름이 X로 시작하고 Active 상태이며 Assistants인 엔트리만을 쿼리한 ResultSet의 결과이다.

	X	active ▾	<input type="radio"/> All <input checked="" type="radio"/> Assistants <input type="radio"/> Executives
#	Name	Active	Title
1	Xiang,Wolfgang S.	1	Assistant Product Manager
2	Xenia,Martin L.	1	Assistant Resources Director
3	Xerxes,William N.	1	Assistant Developer
4	Xenia,Samantha X.	1	Assistant Administrator
5	Xenia,Roger X.	1	Assistant Resources Director

이 장에서는 Zen 테이블의 다음과 같은 속성들에 대하여 설명한다:

- Zen 페이지에서의 테이블 배치
- Zen 테이블에 대한 데이터 소스 설정
- 테이블 쿼리에 대한 파라미터 정의
- 필터와 링크를 포함한 컬럼 속성들 설정
- Zen 테이블이 가질 수 있는 스타일 프로퍼티 설정
- 행과 컬럼에 대한 데이터 형식 정의
- 복수페이지 테이블에 대한 스냅샷 모드 및 테이블 새로고침(refresh) 정의
- Zen 테이블에서의 사용자 인터페이스 정의
- 테이블 새로고침 수행 절차 및 요청 방법

### <tablePane>

<tablePane>은 다목적 %ZEN.Component.tablePane 클래스의 XML 프로젝션이다. Zen 페이지에 테이블을 정의하려면, <tablePane> 컴포넌트를 페이지 클래스 XData Contents 블록에 위치 시킨다.

이 세션에서는, 테이블을 정의하기 위해 Zen이 제공하는 다양한 컴포넌트들과 관련 클래스들에 대해서 설명한다. 다음은 이러한 클래스들을 XData Contents에서 표현할 때 사용하는 XML 엘리먼트들에 대한 설명이며 그 중 가장 중요한 것은 <tablePane> 이다:

- <tablePane> SQL 쿼리에 기반한 HTML 테이블을 작성한다. ResultSet의 각 행은 테이블에 행으로 매핑된다. <tablePane> 는 다음과 같은 엘리먼트들을 갖을 수 있다.
  - <parameter> 각 <parameter> 엘리먼트는 <tablePane> 쿼리를 구성하는데 필요한 파라미터를 제공한다.
  - <column> 각 <column> 엘리먼트는 결과 테이블의 특정 컬럼에 대한 배치, 스타일 및 기능의 세부 사항을 정의한다. <column> 엘리먼트는 ResultSet의 모든 컬럼이 표시될 때는 선택적으로 사용되지만 <tablePane>의 출력할 컬럼을 지정하는 경우에는 <column> 엘리먼트는 반드시 정의되어져야 한다.
  - <condition> 각 <condition> 엘리먼트는 테이블의 행과 cell에 적용하는 데이터 전용의 세부 사항을 정의한다. 예를 들면, 빨간 색으로 오류 조건을 나타내는 것과 같이 특정한 값을 가지는 cell의 배경색을 정의할 수도 있다. 이러한 값을 가지는 특정 cell은 테이블이 새로이 고쳐질 때마다 바꿔질 수도 있는데, Zen은 이러한 조건들을 매번 확인하고 모든 cell에 적절한 색을 지정한다.
- <tableNavigator> 복수의 페이지를 갖는 테이블의 페이지간 이동에 필요한 기본 버튼들을 자동적으로 제공한다.
- <tableNavigatorBar> <tableNavigator> 엘리먼트보다 추가의 버튼들을 제공하여 보다 규모가 크고 페이지수가 많은 테이블 제어에 도움을 준다.

<tablePane>에는 다음과 같은 범용 속성들이 있다.

속성	설명
dataSource	<p>%ResultSet 으로부터 출력할 컬럼과 그 순서를 명시한다. 사용 가능할 값으로는</p> <ul style="list-style-type: none"> <li>“query” – 왼쪽에서 오른쪽 순서로 쿼리에 설정된 모든 컬럼을 출력한다.</li> <li>“columns” – &lt;tablePane&gt;의 &lt;column&gt; 엔트리에 정의된 컬럼들만 출력한다. 왼쪽에서 오른쪽 순서.</li> </ul> <p>&lt;tablePane&gt;에서 dataSource를 생략하면, 디폴트 값은 “query”이다. &lt;column&gt; 엔트리가 정의되어 있으면, dataSource 값은 무시되고, “columns” 값이 사용된다.</p>
initialExecute	<p>initialExecute 값이 true 면, &lt;tablePane&gt;에서 정의된 쿼리는 테이블이 처음 출력될 때 실행된다. 그렇지 않은 경우, &lt;tablePane&gt;은 요청 시에만 쿼리를 실행한다. 디폴트 값은 true이다.</p> <p>initialExecute의 데이터 타입은 %ZEN.Datatype.boolean 인데 XData Contents 블록에서는 true 또는 false 값을, 서버상의 코드에서는 1 또는 0, 그리고 클라이언트 코드에서는 true 또는 false의 값을 갖는다.</p>

<tablePane>에는 배치, 스타일, 기능을 구성하는데 필요한 추가적인 많은 속성들을 갖고 있으며 다른 엘리먼트들을 소개하면서 추가적으로 설명한다.

## 데이터 소스

<tablePane> 엘리먼트는 다음 중 하나의 방법으로 반드시 테이블의 데이터 소스를 지정해야 한다:

- SQL 쿼리문
- SQL 쿼리를 생성에 필요한 기술
- 미리 정의되어 있는 클래스 쿼리
- ResultSet을 리턴하는 콜백(CallBack) 메소드
- 런타임에 테이블에 대한 데이터 소스를 동적으로 변경

위의 각 방법들에 대해 설명한다. 어떤 방법이든 모두 maxRows 속성을 갖는다:

속성	설명
maxRows	리턴되는 행의 최대값. 일반적으로 출력할 행의 최대수를 의미하며, 스냅샷(snapshot) 테이블의 경우, maxRows는 스냅샷(snapshot)의 최대 크기를, pageSize는 페이지당 출력할 행의 수를 의미한다. maxRows의 디폴트 값은 100이다.

### ➤ SQL 쿼리의 명시

<tablePane>의 *sql* 속성을 통해 완전한 SQL 문을 설정할 수 있다. Zen은 이 SQL 문을 실행하여 테이블의 내용을 구성한다. <radioSet>, <select>, <dataListBox>, <dataCombo>,

<repeatingGroup> 등도 동일한 *sql* 속성을 지원한다.

다음은 간단한 SQL 서술문의 예이다.

```
<tablePane id="table"
    sql="SELECT ID,Name FROM MyApp.Employees
        WHERE Name %STARTSWITH ? ORDER BY Name"
/>>
```

<tablePane>의 <parameter> 엘리먼트를 사용하여 입력 파라미터 값을 정의할 수도 있다.

<tablePane> 속성 *sql* 은 %ZEN.Component.tablePane 프로퍼티 *sql* 의 XML 프로젝션이며, %ZEN.Component.tablePane의 다른 프로퍼티들과는 달리, 실행시 클라이언트에서 설정될 수 없고, XData Contents 에서만 설정 가능하다. 이는 *sql*이 암호화 속성이기 때문이다. *sql* 속성 값은 클라이언트에 보내질 때 현재의 세션 키를 사용, 암호화되며 서버에서 자동적으로 복호화된다.

이는 클라이언트 브라우저에서 페이지 소스를 보는 경우, 요청된 SQL 문이 보이지 않게 하여 쿼리 구성으로부터 유출될 수 있는 클라인언트 로직을 보호하기 위함이다. 보안상의 이유로, 쿼리 작업을 항상 서버에만 국한되어야 한다.

*sql* 속성 값으로 XML 특수 문자를 사용할 수 없다. 예를 들면, ‘-보다 작다’라는 의미의 심볼 < 대신, 아래와 같이 XML 엔티티인 &lt; 를 사용하여야 한다:

```
sql="select * from infonet_daten.abopos where lieferadresse=? and status&lt;9"
```

다음 테이블은 *sql* 속성값에 사용되면 문제가 되는 XML 특수 문자와 대체하여야 할 XML 엔티티 목록이다.

#### SQL 속성 값에 사용되는 XML 엔티티

문자	XML 엔티티	설명
>	&gt;	오른쪽 각 괄호 또는 “보다 크다”는 심볼
<	&lt;	왼쪽 각 괄호 또는 “보다 작다”는 심볼
&	&amp;	&
'	&apos;	작은 따옴표 또는 아포스트로피. 작은따옴표에 들어가는 문자열에는 ‘을 대신하는 엔티티인 &apos; 가 필요함
"	&quot;	큰따옴표. 큰따옴표에 들어가는 문자열에는 “을 대신하는 엔티티인 &quot; 가 필요함

#### ➤ SQL 쿼리 생성

또한 <tablePane> 는 간단한 설명을 기반 SQL 쿼리를 자동 생성할 수 있는 속성들을 지원한다.

기본적으로 이러한 속성들은, Zen이 쿼리를 기반으로 콜백 메소드를 생성한다는 점을 제외하고는 콜백(CallBack) 메소드를 사용하는 것과 유사하다. <dataListBox> 와 <dataCombo> 또한 이 속성을 지원한다.

속성	설명
<i>groupByClause</i>	(옵션) "Year,State"과 같은 SQL GROUP BY 절. <i>groupByClause</i> 값은 문자열 또는 Zen의 #()# 실시간 표현식을 사용한다.
<i>orderByClause</i>	(옵션) "Name,State"과 같은 SQL ORDER BY 절. 이 값이 정의되어 있지 않은 경우 컬럼 헤더를 클릭하면, 다음 쿼리는 사용자 선택에 기반한 ORDER BY 절을 갖는다. <i>orderByClause</i> 값은 문자열이나 Zen의 #()# 실시간 표현식을 사용한다.
<i>tableName</i>	검색할 데이터를 갖고 있는 SQL 테이블 이름. 이 값은 생성된 쿼리의 FROM 절에 사용된다. <i>tableName</i> 값은 문자열이나 Zen의 #()# 실시간 표현식을 사용한다.
<i>whereClause</i>	(옵션) "Name='Elvis'"과 같은 SQL WHERE 절. 이 값이 정의되어 있지 않은 경우, 컬럼 필터가 이 테이블에 정의되어 있으면, WHERE 절은 현재의 필터 값을 기반으로 생성된다. <i>whereClause</i> 값은 문자열이나 Zen의 #()# 실시간 표현식을 사용한다.

<tablePane> 으로 SQL 쿼리를 생성하기 위해 다음과 같은 작업을 수행 한다:

1. *tableName* 속성 값을 정의한다.
  2. *sql*, *queryClass*, *queryName*, 또는 *OnCreateResultSet*에 대한 값을 설정하지 않는다. 이 속성들은 여기서 설명하는 속성들에 우선하기 때문이다.
- 위의 두 조건을 만족하는 경우 Zen은 “columns”의 값을 데이터 소스로 인식하게 된다.
3. <tablePane> 내에 <column> 엘리먼트를 제공하여, 컬럼명을 정의한다. 최소한 하나의 컬럼은 정의되어야 하며, 그렇지 않은 경우 “Missing SELECT list” 오류가 발생한다.
  4. <tablePane> 내에 <parameter> 엘리먼트를 제공하여 쿼리 파라미터를 추가할 수 있다.
  5. <tablePane> 속성 *groupByClause*, *orderByClause*, *whereClause* 를 설정하여 생성된 쿼리에 절을 추가할 수 있다.

다음은 간단한 예이다:

```
<tablePane id="table"
    tableName="MyApp.Employee">
    <column colName="ID" hidden="true"/>
    <column colName="Name"/>
</tablePane>
```

위의 <tablePane> 예는 다음과 유사한 SQL 쿼리문을 생성한다:

```
SELECT ID,Name FROM MyApp.Employee
```

Zen은 이 쿼리를 실행하여 테이블의 ResultSet을 제공한다. 위의 예에서 ID 컬럼이 hidden 으로

설정되었는데, 이는 그 값이 (쿼리 조건과 실행에 따라) ResultSet에 포함은 되지만 출력은 되지 않는다는 의미이다.

**참조:** 생성된 SQL 쿼리는 컬럼 필터를 가진 테이블에 유용하게 사용된다.

#### ➤ 클래스 쿼리 참조하기

<tablePane> 은 기존의 클래스 쿼리를 참조하여, %ResultSet 오브젝트를 얻을 수도 있다. 다음 속성들은 이러한 기능을 지원하며, <radioSet>, <select>, <dataListBox>, <dataCombo>, <repeatingGroup> 또한 이 속성들을 지원한다.

속성	설명
<i>queryClass</i>	쿼리를 포함하고 있는 클래스명. 반드시 <i>queryName</i> 값도 같이 주어야 한다.
<i>queryName</i>	<tablePane>을 위해 %ResultSet를 제공하는 클래스 쿼리명. 반드시 <i>queryClass</i> 값도 같이 주어야 한다.

<parameter> 엘리먼트를 <tablePane> 내에 위치시켜 쿼리의 입력 파라미터 값을 제공할 수도 있다. 예:

```
<tablePane id="table"
           queryClass="MyApp.Employee"
           queryName="ListEmployees">
    <parameter value="Sales"/>
    <parameter value="NEW YORK"/>
</tablePane>
```

#### ➤ 콜백(CallBack) 메소드 사용하기

<tablePane>는 %ResultSet 오브젝트를 사용하는 테이블을 생성한다. 다음은 이 기능에 필요한 <tablePane>의 속성들이다. 이 속성들은 <dataListBox>, <dataCombo>에서도 동일하게 지원한다.

속성	설명
<i>OnCreateResultSet</i>	서버상의 Zen 페이지 클래스에 있는 콜백 메소드명. 이 메소드는 %ResultSet 오브젝트를 정의하고 필요한 %ZEN.Auxiliary.QueryInfo 프로퍼티들을 설정한다. 테이블을 작성할 때마다, Zen은 이 메소드를 호출하여, 자동적으로 아래의 파라미터들을 전달한다: <ul style="list-style-type: none"><li>• %Status – 메소드 상태에 대한 출력 파라미터</li><li>• %ZEN.Auxiliary.QueryInfo – &lt;tablePane&gt;에 대한 QueryInfo 오브젝트</li></ul> 콜백 메소드는 반드시 %ResultSet 오브젝트를 리턴해야 한다. 다음은 유효한 메소드 서명이다.

	<p style="text-align: center;"><b>Method      CreateRS(Output      pSC      As      %Status,      pInfo As %ZEN.Auxiliary.QueryInfo) As %ResultSet</b></p>
	<p>위의 메소드를 콜백 메소드로 사용하려면, &lt;tablePane&gt;에 대하여 <b>OnCreateResultSet="CreateRS"</b> 를 설정한다.</p> <p>일단 정의되면, <i>OnCreateResultSet</i>는 &lt;tablePane&gt;에 데이터를 제공하기 위한 다른 설정들에 대해 우선하게 된다.</p>
<i>OnExecuteResultSet</i>	<p>서버상의 Zen 페이지 클래스에 있는 콜백 메소드명. 이 메소드로 <i>OnCreateResultSet</i> 콜백에 의해 리턴되는 %ResultSet 오브젝트를 실행한다. Zen은 <i>OnCreateResultSet</i> 콜백을 호출한 후 자동적으로 <i>OnExecuteResultSet</i> 콜백을 호출하여, 아래의 파라미터들을 전달한다:</p> <ul style="list-style-type: none"> <li>• %ResultSet – <i>OnCreateResultSet</i> 으로부터 생성된 결과 집합</li> <li>• %Status – 메소드 상태에 대한 출력 파라미터</li> <li>• %ZEN.Auxiliary.QueryInfo – &lt;tablePane&gt;에 대한 QueryInfo 오브젝트</li> </ul> <p>선택적으로, <i>OnCreateResultSet</i> 콜백의 끝에 QueryInfo <i>queryExecuted</i> 프로퍼티를 true 로 설정하여 <i>OnExecuteResultSet</i> 콜백의 호출을 금지시킬 수 있다.</p> <p><i>OnExecuteResultSet</i> 콜백은 %ResultSet이 실행되었는지 여부를 알려주는 %Boolean을 리턴한다. 다음은 유효한 메소드 서명이 된다.</p> <p style="text-align: center;"><b>Method ExecuteRS(myRS As %ResultSet, Output pSC As %Status, pInfo As %ZEN.Auxiliary.QueryInfo) As %Boolean</b></p>
	<p>위의 메소드를 콜백 메소드로 사용하려면, &lt;tablePane&gt;에 대하여 <b>OnExecuteResultSet="ExecuteRS"</b> 를 설정한다.</p>
<i>showQuery</i>	<p><i>showQuery</i> 는 테이블 생성에 <i>OnCreateResultSet</i> 을 사용하는 경우와 쿼리의 텍스트를 포함하기 위해 이 콜백이 QueryInfo 오브젝트의 <i>queryText</i> 프로퍼티를 설정하는 경우에만 적용된다. SQL 쿼리를 생성하기 위해 콜백 메소드를 사용하는 여러 컴포넌트 중에서, &lt;tablePane&gt; 과 &lt;dataCombo&gt; 만이 <i>showQuery</i> 속성을 지원한다.</p> <p><i>showQuery</i> 가 true 로 설정되면, Zen 페이지는 &lt;tablePane&gt; 또는 &lt;dataCombo&gt; 컴포넌트에 쿼리 결과를 제공하는데 사용된 SQL 쿼리문을 보여준다. 이 속성은 개발 시, 문제 해결 목적에서 효과적으로 사용할 수 있다. <i>showQuery</i> 의 디폴트 값은 false 이다.</p> <p><i>showQuery</i> 는 기본 데이터 타입으로 %ZEN.Datatype.boolean 이 있으며, 이 값은 XData Content 에서는 true 또는 false, 서버상의 코드에서는 1 또는 0, 그리고 클라이언트에서는 true 또는 false 로 설정된다.</p> <p><i>showQuery</i> 값은 문자열이나 Zen #()# 실시간 표현식을 가질 수 있다.</p>

다음은 콜백 메소드를 사용하여, **tablePane**에 대한 %ResultSet를 생성하는 예제이다. 콜백 메소드는 **tablePane**의 포로퍼티인 *sortOrder* 와 *sortColumn* 의 현재 값에 대응하는 동적 SQL 문을 구성한다. **tablePane**은 이 값을 자동적으로 입력, %ZEN.Auxiliary.QueryInfo 오브젝트의 상응하는 프로퍼티로서 콜백 메소드에 제공한다.

또한 메소드가 완료되기 전, 생성된 SQL 문을 어떻게 **QueryInfo**의 *queryText* 프로퍼티에 설정하는지를 보여준다. <tablePane>의 *showQuery* 값이 true 면, 생성된 쿼리문과 함께 테이블을 출력한다.

```
Method CreateRS(Output tSC As %Status, pInfo As %ZEN.Auxiliary.QueryInfo) As %ResultSet
{
    Set tRS = ""
    Set tSC = $$$OK
    Set tSELECT = "ID,Name,Title"
    Set tFROM = "MyApp.Employee"
    Set tORDERBY = pInfo.sortColumn
    Set tSORTORDER = pInfo.sortOrder
    Set tWHERE = ""
    // Build WHERE clause based on filters
    If ($GET(pInfo.filters("Name"))!="") {
        Set tWHERE = tWHERE _ $SELECT(tWHERE=""",1:" AND ")_
                    "Name %STARTSWITH "" _ pInfo.filters("Name") _ """
    }
    If ($GET(pInfo.filters("Title"))!="") {
        Set tWHERE = tWHERE _ $SELECT(tWHERE=""",1:" AND ")_
                    "Title %STARTSWITH "" _ pInfo.filters("Title") _ """
    }
    Set sql = "SELECT " _ tSELECT _ " FROM " _ tFROM
    Set:tWHERE'="" sql = sql _ " WHERE " _tWHERE
    Set:tORDERBY'="" sql =
        sql _ " ORDER BY " _tORDERBY _ $SELECT(tSORTORDER="desc": " desc",1:"")
    Set tRS = ##class(%ResultSet).%New()
    Set tSC = tRS.Prepare(sql)
    Set pInfo.queryText = sql
    Quit tRS
}
```

다음 테이블에서는 위의 두 콜백 메소드의 서명에 사용되는 %ZEN.Auxiliary.QueryInfo 오브젝트의 프로퍼티에 대해 설명한다.

**tablePane**과 마찬가지로, **dataCombo**, **dataListBox**, **repeatingGroup** 컴포넌트 또한 이 콜백 메소드를 사용하여 %ResultSet 으로부터의 컴포넌트를 생성한다. **QueryInfo** 오브젝트에 있는 몇몇 프로퍼티들은 **tablePane** 쿼리에만 적용되며 **dataCombo**, **dataListBox**, **repeatingGroup** 은 **columns**, **filters**, 및 **sorting**을 포함하여 테이블-전용 프로퍼티들은 무시한다.

오직 **tablePane** 과 **dataCombo** 만이 *queryText* 프로퍼티를 지원한다.

## QueryInfo 프로퍼티

프로퍼티	설명
<i>columnExpression</i>	<tablePane>의 각 <column>에서의 <i>colExpression</i> 값. <i>columnExpression</i> 은 이(러한) 값들을 <column> <i>colName</i> 아래의 다차원 배열로 조직한다.
<i>columns</i>	<tablePane>의 각 <column>에서의 <i>colName</i> 값. <i>columns</i> 은 이 값들을 컬럼번호 (1부터 시작) 아래의 다차원 배열로 조직한다.
<i>filterOps</i>	<tablePane>의 각 <column>에서의 <i>filterOp</i> 값. <i>filterOps</i> 는 이 값을 <column> <i>colName</i> 아래의 다차원 배열로 조직한다.
<i>filters</i>	<tablePane>의 각 <column>에서의 <i>filterValue</i> 값. 필터는 이 값을 <column> <i>colName</i> 아래의 다차원 배열로 조직한다.
<i>filterTypes</i>	<tablePane>의 각 <column>에서(의) <i>filterType</i> 값. <i>filterTypes</i> 는 이 값을 <column> <i>colName</i> 아래의 다차원 배열로 조직한다.
<i>groupByClause</i>	<tablePane>의 <i>groupByClause</i> 값 (제공된 경우)
<i>orderByClause</i>	<tablePane>의 <i>orderByClause</i> 값 (제공된 경우)
<i>parms</i>	파라미터 번호 아래로(1부터 시작) 만들어진 다차원 배열. 이 배열은 <tablePane> 내의 <parameter> 엘리먼트에 의해 제공되는 입력 값을 갖게된다.
<i>queryExecuted</i>	<i>OnCreateResultSet</i> 에 의해 식별되는 메소드에서 이 속성을 true로 설정하면 새로 생성된 %ResultSet 가 이미 실행되어 <i>OnExecuteResultSet</i> 에 의해 식별된 메소드를 호출될 필요가 없게 된다. 디폴트 값은 false이다.
<i>queryText</i>	<i>OnCreateResultSet</i> 에 의해 식별된 메소드는 이 값을 실제의 쿼리 텍스트에 설정하여, <i>showQuery</i> 가 true이면 나타나도록 할 수 있다.
<i>rowCount</i>	이 값이 콜백에 의해 설정되면, 반환 시 <i>rowCount</i> 프로퍼티는 쿼리로부터 반환되는 행의 총수를 갖게된다. 쿼리가 실행된 후, <i>rowCount</i> 는 <i>rows</i> 와 달라질 수 있다.  <i>rowCount</i> 는 숫자가 아니고 스트링 타입으로 그 값은 "" 또는 "100+" 형태가 될 것이다. 100 이상의 행 숫자는 "100+"로 표시된다. JavaScript에서 <i>rowCount</i> 를 테스트하는 경우, 만약 숫자 값으로 전환하려면, base 10에 대한 <i>parseInt</i> 를 사용한다:  <i>rowCount = parseInt(rowCount,10);</i>
<i>rows</i>	요청된 행 수. 테이블에서는, 이 값이 <i>maxRows</i> 값이다.
<i>sortColumn</i>	사용자가 선택한 현재의 정렬 컬럼의 <i>colName</i>
<i>sortOrder</i>	"asc" 또는 "desc"로 테이블의 현재 정렬 순서 (보통 사용자 클릭으로 정해짐)

<i>tableName</i>	<tablePane> 의 <i>tableName</i> 값
<i>whereClause</i>	<tablePane> 의 <i>whereClause</i> 값 (제공된 경우)

## 쿼리 파라미터

SQL 쿼리를 사용하여 Zen 테이블에 대한 데이터를 생성할 때, 쿼리 입력 파라미터의 값을 제공하기 위해 ? 문자를 사용하여야 하는 경우가 발생한다. Zen은 이 입력 파라미터를 정의하기 위하여 <tablePane> 내에 <parameter> 엘리먼트를 사용한다. <radioSet>, <select>, <dataListBox>, <dataCombo>, <repeatingGroup> 도 쿼리에 대하여 동일한 <parameter> 를 사용할 수 있다.

속성	설명
<i>value</i>	<p>파라미터 값 정의:</p> <p>&lt;parameter value="Here is my value!" /&gt;</p> <p>파라미터(에 주어지는) 값은 문자열 또는 Zen의 #()# 실시간 표현식을 가질 수 있다.</p>

“데이터 소스” 섹션에서는 다음 클래스 쿼리 예제를 비롯하여 <parameter> 엘리먼트를 사용하는 여러 예제를 제공하고 있다:

```
<tablePane id="table"
    queryClass="MyApp.Employee"
    queryName="ListEmployees">
<parameter value="Sales"/>
<parameter value="NEW YORK"/>
</tablePane>
```

%ZEN.Component.tablePane을 프로그램에서 사용하는 경우, <parameter> 요소를 list 컬렉션인 tablePane *parameters* 프로퍼티의 멤버로 취급한다. <tablePane>의 각 <parameter>는 순서 위치 1,2,3 등으로 결합된 tablePane의 *parameters* 컬렉션의 멤버가 된다.

tablePane 오브젝트의 *parameters* 컬렉션 프로퍼티에 값을 설정함으로써 실행 시, 클라이언트에서 쿼리 파라미터 값을 변경할 수 있다. 다음은 첫번째 파라미터값을 Finance로 변경하여, 서버에서 쿼리를 재실행하고, tablePane의 내용을 새 값으로 업데이트하여 출력하는 예제이다:

```
Method changeParams() [ Language = javascript ]
{
    // find the tablePane component
    var table = this.getComponentById('table');
    table.setProperty('parameters',1,'Finance');
}
```

## 테이블 컬럼

“데이터 소스” 섹션에서는 <tablePane> 이 SQL 쿼리를 기반으로한 HTML 테이블을 작성하는 것에 대하여 설명하고 있다. 쿼리 결과 집합의 각 행은 테이블에 행으로 출력된다. <tablePane>은 하나

이상의 <column> 엘리먼트를 가지는데, 이 엘리먼트들이 쿼리를 통해 선택된 컬럼들로서 테이블에 출력될 뿐만 아니라 테이블의 각 컬럼에 대한 배치, 스타일, 기능등을 정의한다.

앞의 “SQL 쿼리 생성” 부분은 다음과 같이 <column> 엘리먼트가 테이블을 생성하는 예제를 제공한다:

```
<tablePane id="table"
  tableName="MyApp.Employee">
  <column colName="ID" />
  <column colName="Name"/>
  <column colName="Title" style="color: blue;" />
</tablePane>
```

<column> 엘리먼트(요소)는 %ZEN.Auxiliary.column 클래스의 XML 프로젝션으로서 다음과 같은 속성들을 지원한다.

속성	설명
<i>cellTitle</i>	컬럼에 있는 cell 대해 툴팁(tooltip) 메시지를 표시하는 텍스트. 이 속성에 설정되는 입력값의 기본 데이터 타입은 %ZEN.Datatype.caption 이다. 이 값은 언어 DOMAIN 파라미터가 Zen 페이지 클래스에 정의되어 있으면 현지 언어로 쉽게 변경된다. 또한 %ZEN.Datatype.caption 데이터 타입은 \$\$\$Text 매크로를 사용하여 <i>cellTitle</i> 프로퍼티 값을 정의할 수 있도록 해 준다.
<i>colExpression</i>	테이블이 자동적으로 SQL 쿼리를 구축하는 경우 이 속성값은 현재 컬럼의 값을 얻는데 필요한 SQL 표현이 된다. 예를 들면 <i>colExpression="Customer-&gt;Name"</i> 과 같이 된다.
<i>colName</i>	현재 컬럼과 링크된 SQL 데이터 컬럼명. 필요하면, 아래와 같이 <i>colName</i> 값에 SQL 함수나 SQL 별칭을 사용할 수도 있다:  <column colName="Salary-1000 As SalaryMinus1000" width="10" filterType="text" filterOp="BETWEEN" filterLabel=" Range (Min,Max):" />  또는  <column colName="max(Salary) As SalaryMax" width="100" filterType="text" filterOp="BETWEEN" filterLabel=" Range (Min,Max):" />  <i>colName</i> 값은 문자열이거나 Zen #()# 실시간 표현식을 가질 수 있다.  만약 동일 <tablePane>에서 <i>colName</i> 값이 중복되면, 두 번째 컬럼은 “(duplicate) <i>colName</i> ” 메시지를 출력하며 <tablePane>은 오작동하게 된다.  특정 컬럼에 대해 <i>colName</i> 이 정의되지 않으면, 이 컬럼은 데이터 값 없이 반환된다. 특히 이 속성은 행에서의 링크를 설정하는데 사용한다.
<i>header</i>	컬럼 헤더용 텍스트. 이 속성값으로 일반적인 텍스트를 사용하지만 기본 데이터 타입은 %ZEN.Datatype.caption 이다. 이 값은 언어 DOMAIN 파라미터가 Zen 페이지 클래스에 정의되어 있으면 현지 언어로 쉽게 변경된다. 또한 %ZEN.Datatype.caption 데이터 타입은 \$\$\$Text 매크로를 사용하여 <i>header</i> 프로퍼티 값을 정의할 수 있도록 해 준다.

	<i>head</i> 값은 문자열 또는 Zen #()# 실시간 표현식을 가질 수 있다.
<i>hidden</i>	<p>이 값이 true 면 이 컬럼은 출력되지 않는다. 디폴트 값은 false.</p> <p>이 속성의 기본 데이터 타입은 %ZEN.Datatype.boolean이며, XData Contents 블록에서는 true 또는 false, 서버상의 코드에서는 1 또는 0, 클라이언트에서는 true 또는 false 값을 갖는다.</p> <p><i>hidden</i> 값은 문자열 또는 Zen #()# 실시간 표현식을 가질 수 있다.</p>
<i>OnDrawCell</i>	<p>(옵션) Zen 페이지 클래스의 서버 쪽 콜백 메소드명. 이 메소드는 &amp;html&lt;&gt; 문법이나 WRITE 명령어를 사용하여, HTML 내용을 컬럼 셀(cell)에 입력한다. Zen은 컬럼을 작성할 때 이 메소드를 호출하며, 아래의 파라미터들에 자동적으로 제공된다:</p> <ul style="list-style-type: none"> <li>● &lt;tablePane&gt; 오브젝트에 대한 포인터</li> <li>● 이 컬럼과 결합된 SQL 데이터 컬럼명. 이 컬럼명으로 데이터를 액세스하기 위해, 아래의 예와 같이, 이 값에 <b>%query</b> 함수를 사용할 수도 있다.</li> <li>● &lt;column&gt; 으로부터의 <i>seed</i> 값</li> </ul> <p>이 콜백은 %Status 데이터 타입을 반환하며, 다음과 같은 메소드 서명을 취한다:</p> <pre>Method DrawYesNo(Cell) (pTable As %ZEN.Component.tablePane, pName As %String, pSeed As %String) As %Status</pre> <p>이 메소드를 콜백으로 사용하려면, &lt;column&gt; 에 OnDrawCell="DrawYesNo" 를 설정한다.</p> <pre>&lt;column colName="WorkDone" header="Complete?" onDrawCell="DrawYesNo" /&gt;</pre> <p>다음의 OnDrawCell 콜백 메소드는 &lt;tablePane&gt; 컬럼에 Yes 또는 No 를 출력하기 위하여 Boolean 값(1 또는 0)을 체크한다. 컬럼값을 검색을 위한 <b>%query</b> 함수 사용에 주목한다:</p> <pre>Method DrawYesNo(pTable As %ZEN.Component.tablePane, pName As %String, pSeed As %String) As %Status { If %query(pName) {Write \$\$\$Text("Yes") } Else { Write \$\$\$Text("No") } Quit \$\$\$OK }</pre>
<i>seed</i>	임의의 값을 <i>OnDrawCell</i> 콜백에 제공할 수 있도록 한다.
<i>style</i>	이 컬럼의 cell(HTML td 엘리먼트)에 적용하는 CSS 스타일 값 (예, "color: red;"). <i>style</i> 값은 문자열 또는 Zen #()# 실시간 표현식을 가질 수 있다.
<i>title</i>	<p>컬럼 헤더로 마우스를 움직일 때 나타나는 툴팁(tooltip) 텍스트.</p> <p>이 속성값으로 일반적인 텍스트를 사용하지만 기본 데이터 타입은 %ZEN.Datatype.caption이다. 이 값은 언어 DOMAIN 파라미터가 Zen 페이지 클래스에 정의되어 있으면 현지 언어로 쉽게 변경된다. 또한 %ZEN.Datatype.caption 데이터 타입은 \$\$\$Text 매크로를 사용하여 <i>style</i> 프로퍼티 값을 정의할 수 있도록 해 준다.</p>
<i>width</i>	일반적으로 Zen 테이블에 대한 HTML <i>style</i> 값은 고정된다. 즉, 각 컬럼은 생성되는 HTML 페이지의 특정 <i>width</i> 값을 갖는다. Zen에서는 이러한 값을 다음과 같이

	<p>정한다:</p> <ul style="list-style-type: none"> <li>각 컬럼에 대하여 <i>width</i> 값을 지정할 수 있다.</li> <li>몇몇 컬럼에 대한 <i>width</i> 값을 지정하지 않은 경우, Zen은 해당 컬럼의 데이터 길이에 비례하여 <i>width</i> 값을 지정한다.</li> <li>테이블의 모든 컬럼에 대해 <i>width</i> 값을 지정하지 않은 경우, Zen은 전체 테이블에 대해 HTML <i>style</i> 값 “auto”를 사용한다.</li> </ul> <p><i>width</i> 값은 문자열 또는 Zen #()# 실시간 표현식을 가질 수 있다.</p>
--	--

<column>은 또한 Zen 테이블 컬럼에 대한 동적 필터링과 링크를 지원하는 속성들을 제공하는데 다음의 섹션에서 상세히 설명한다:

- 컬럼 필터
- 컬럼 링크

마지막으로 사용자 선택에 따라 컬럼 배치를 동적으로 변경 가능하도록 <tablePane>은 테이블의 모든 컬럼에 적용하는 속성들을 제공한다. 이에 대한 자세한 설명은 다음 섹션에서 제공된다:

- 테이블 정렬
- 행과 컬럼의 선택

%ZEN.Component.tablePane을 이용하여 프로그램을 작성하는 경우, <column> 엘리먼트를 %ZEN.Auxiliary.column 오브젝트의 리스트 컬렉션인 tablePane *column* 프로퍼티의 멤버로서 사용한다. <tablePane>의 각 <column>은 순서 위치(표시)로 1, 2, 3 등과 결합된 tablePane의 *column* 컬렉션의 멤버가 된다.

## 테이블 스타일

<tablePane>은 테이블 스타일을 제어하기 위해 다음과 같은 속성들을 사용한다.

속성	설명
<i>caption</i>	테이블의 캡션 텍스트. 이 속성값으로 일반적인 텍스트를 사용하지만 기본 데이터 타입은 %ZEN.Datatype.caption이다. 이 값은 언어 DOMAIN 파라미터가 Zen 페이지 클래스에 정의되어 있으면 현지 언어로 쉽게 변경된다. 또한 %ZEN.Datatype.caption 데이터 타입은 \$\$\$Text 매크로를 사용하여 <i>caption</i> 프로퍼티 값을 정의할 수 있도록 해 준다.
<i>extraColumnWidth</i>	여러 행을 선택하거나, 행 수가 tablePane에 표시되는 경우, 여분의 컬럼을 위한 HTML의 폭 (width). 디폴트 값은 30이다.
<i>fixedHeaders</i>	이 값이 true면, 테이블 헤더는 테이블 본문을 스크롤 할 때, 고정된 위치에 계속 머물게 된다. 기본은 false이다. 이 속성의 기본 데이터 타입은 %ZEN.Datatype.boolean으로 그 값은 XData Contents 블록에서는 true 또는 false, 서버쪽 코드에서는 1 또는 0,

	클라이언트에서는 true 또는 false 이다.
<i>bodyHeight</i>	<i>fixedHeaders</i> 가 true 면, <i>bodyHeight</i> 는 테이블의 본문 높이를 설정하는 HTML 길이 값을 제공한다. <i>bodyHeight</i> 의 디폴트 값은 “20.0em” 이다.
<i>nowrap</i>	이 값이 true 면, 테이블 cell의 헤더들의 자동 줄바꿈이 허용되지 않는다. 디폴트 값은 false 이다. 이 속성의 기본 데이터 타입은 %ZEN.Datatype.boolean 이며, 이 값은 XData Contents 블록에서는 true 또는 false, 서버쪽 코드에서는 1 또는 0, 클라이언트에서는 true 또는 false 이다.
<i>showRowNumbers</i>	이 값이 true 면, tablePane 왼쪽 컬럼에 행 번호가 표시된다. 디폴트 값은 false 이다. 이 속성의 기본 데이터 타입은 %ZEN.Datatype.boolean 이며, 그 값은 XData Contents 블록에서는 true 또는 false, 서버쪽 코드에서는 1 또는 0, 클라이언트에서는 true 또는 false 이다.
<i>showValueInTooltip</i>	이 값이 true 면, 테이블의 셀(cell)에 대한 툴팁(HTML <i>title</i> 속성) 값은 해당 cell 의 현재 값으로 구성된다. 디폴트 값은 false 이다. 이 속성의 기본 데이터 타입은 %ZEN.Datatype.boolean 이며, 그 값은 XData Contents 블록에서는 true 또는 false, 서버쪽 코드에서는 1 또는 0, 클라이언트에서는 true 또는 false 이다.
<i>showZebra</i>	이 값이 true 면, 얼룩말 줄무늬를 사용(어둡고 밝은 행이 교대로 그려지며 tablePane 을 구성한다. 디폴트 값은 false 이다. 이 속성의 기본 데이터 타입은 %ZEN.Datatype.boolean 이며, 그 값은 XData Contents 블록에서는 true 또는 false, 서버쪽 코드에서는 1 또는 0, 클라이언트에서는 true 또는 false 이다.

### 행과 cell 에 대한 조건부 스타일

<tablePane>은 하나 이상의 <condition> 엘리먼트를 사용할 수 있는데 각 <condition> 엘리먼트는 주어진 행의 값을 기반으로 행이나 각 셀(cell)의 스타일을 제어하는 간단한 표현식이다:

```
<tablePane id="table"
    sql="SELECT Name,Home_State FROM MyApp.Employee">
<condition colName="Name"
    predicate="STARTSWITH"
    value="A"
    rowStyle="background: plum;"/>
</tablePane>
```

위의 예에서, “A” 로 시작하는 Name 컬럼의 값을 갖는 모든 행은 plum(짙은 보라색) 배경으로 표시된다.

일반적으로, 조건부 스타일 기법은 특별한 값을 갖는 행이나 셀(cell)을 강조하기 위해 사용한다. 조건을 추가하는 것은 테이블 출력 작업에 부담을 주기 때문에 선별적으로 사용하는 것이 좋다.

<condition> 엘리먼트는 다음의 속성들을 지원한다.

속성	설명
<i>cellStyle</i>	행에 대한 조건이 만족되면, 대상 컬럼의 셀(cell)에 적용되는 CSS 스타일. 예: "color: red;"
<i>colName</i>	필수 속성. <condition>에 의해 평가되는 데이터 값을 제공하는 컬럼명. <i>colName</i> 값은 문자열 또는 Zen #()# 실시간 표현식을 가질 수 있다.
<i>predicate</i>	조건을 평가하는데 사용하는 논리 연산자. <i>predicate</i> 는 다음의 비교 연산자중 하나이다: "", "GT", "EQ", "LT", "NEQ", "GTEQ", "LTEQ", "EXTEQ", "STARTSWITH", "CONTAINS". 디폴트 <i>predicate</i> 는 "EQ" 이다. 각 연산자에 대한 상세 정보는 "<condition> predicate 값" 테이블을 참조한다.
<i>rowStyle</i>	이 조건을 만족하는 행에 적용하는 CSS 스타일. 예: "font-weight: bold;"
<i>targetCol</i>	<i>cellStyle</i> 이 적용되는 컬럼명. 설정되지 않았으면, <i>colName</i> 값이 사용된다. <i>targetCol</i> 값은 문자열 또는 Zen #()# 실시간 표현식을 가질 수 있다.
<i>value</i>	<i>colName</i> 에서 식별되는 컬럼 값과 비교되는 문자 값. {} 안에 있으면(예, "{Title}") <i>value</i> 는 다른 컬럼명으로 처리되어 해당 컬럼에 있는 값이 사용된다. <i>value</i> 값은 문자열 또는 Zen #()# 실시간 표현식을 가질 수 있다.

테이블이 출력되면, <tablePane>의 모든 <condition> 엘리먼트들은 테이블의 각 행에 대해 개별적으로 평가된다. <condition>이 true 인 경우, 그 조건에 대한 *rowStyle* 또는 *cellStyle* 은 각각 행 또는 cell에 적용된다.

<condition> *predicate* 속성은 다음과 같은 값을 갖는다.

#### <condition> predicate 값

Predicate	설명
CONTAINS	<i>colName</i> 으로 식별되는 컬럼값이 <i>value</i> 에 설정된 값을 갖고 있으면 true 이다.
EQ	<i>colName</i> 으로 식별되는 컬럼값이 <i>value</i> 에 설정된 값과 같으면 true 이다.
EXTEQ	<i>colName</i> 으로 식별되는 컬럼의 파일명이 <i>value</i> 에 설정된 파일 확장명과 같으면 true 이다.
GT	<i>colName</i> 으로 식별되는 컬럼값이 <i>value</i> 에 설정된 값보다 클 경우 true 이다.
GTEQ	<i>colName</i> 으로 식별되는 컬럼값이 <i>value</i> 에 설정된 값과 같거나 클 경우 true 이다.
LT	<i>colName</i> 으로 식별되는 컬럼값이 <i>value</i> 에 설정된 값보다 작을 경우 true 이다.
LTEQ	<i>colName</i> 으로 식별되는 컬럼값이 <i>value</i> 에 설정된 값과 같거나 작을 경우 true 이다.
NEQ	<i>colName</i> 으로 식별되는 컬럼값이 <i>value</i> 에 설정된 값과 다르면 true 이다.

STARTSWITH	<code>colName</code> 으로 식별되는 컬럼값이 <code>value</code> 에 설정된 값으로 시작하면 true 이다.
------------	--

ZEN.Component.tablePane을 프로그램에서 사용하는 경우, <condition> 엘리먼트를 %ZEN.Auxiliary.condition 오브젝트의 리스트 컬렉션인 **tablePane** 의 *conditions* 프로퍼티의 멤버로 사용한다. <tablePane>의 각 <condition>은 해당 *conditions* 컬렉션의 멤버가 되며, 순서 위치 1, 2, 3 등과 결합한다.

### 스냅샷(snapshot) 모드

<tablePane> 을 스냅샷(snapshot) 모드로 실행할 수 있다. 이 모드에서, Zen은 테이블 쿼리를 한 번 실행하고 이 결과를 서버의 임시 위치에 복사하여 둔다. 이후 화면 새로고침을 실행하면 해당 쿼리를 다시 수행하는 대신, 임시 위치에 있는 데이터를 가져다 출력한다. Zen은 자동적으로 임시 스냅샷(snapshot) 데이터의 생성과 주기를 관리한다. 스냅샷(snapshot) 모드는 특히 여러 페이지의 테이블이나 사용자가 정렬할 수 있는 테이블에 특히 유용하다.

<tablePane> 엘리먼트는 스냅샷(snapshot) 모드를 위한 다음 속성들을 지원한다.

속성	설명
<code>useSnapshot</code>	이 값이 true 면, 이 <tablePane>은 스냅샷(snapshot) 모드에 있게 된다. 즉, 데이터를 읽을 때마다 그 데이터는 서버의 임시 위치로 복사된다. 페이징과 정렬 작업은 쿼리를 다시 실행하는 대신 이 스냅샷(snapshot) 데이터를 사용하게 된다. 디플트 값은 false 이다.  이 속성의 기본 데이터 타입은 %ZEN.Datatype.boolean 이며, 이 값은 XData Contents 블록에서는 true 또는 false, 서버쪽 코드에서는 1 또는 0, 클라이언트에서는 true 또는 false로 가진다.
<code>pageSize</code>	스냅샷(snapshot) 테이블에서 이 속성은 데이터를 여러 페이지에 걸쳐 출력하는 경우 한 페이지에 출력되는 행수이다. 디플트 값 0 은 모든 데이터를 첫 페이지에 출력한다는 의미이며, 테이블이 스냅샷(snapshot) 모드에 있을 때만 0 이외의 값으로 설정된다.  <tablePane> 쿼리에서 리턴되는 모든 행을 의미하는 <i>maxRows</i> 와 비교해 본다.

본 장에서 설명하는 모든 쿼리 관계 메커니즘은 스냅샷(snapshot) 모드 또는 direct(non-snapshot) 모드로 사용할 수 있다. 다음은 SQL 쿼리를 스냅샷(snapshot) 모드로 사용하는 예이다:

```
<tablePane id="table"
    sql="SELECT Name,Home_State FROM MyApp.Employee"
    useSnapshot="true"
    pageSize="25"
/>
```

다음의 %ZEN.Component.tablePane 프로퍼티는 <tablePane> 정의에서 XML 속성으로 사용되지 않지만, 페이지가 일단 생성되면 스냅샷(snapshot) 테이블에 유용하게 사용할 수 있다.

속성	설명
<i>clearSnapshot</i>	Zen에서 저장된 스냅샷(snapshot) 사용을 중지하고 테이블 쿼리를 강제로 다시 실행하기 위해 true로 설정하는 실시간 플래그
<i>currPage</i>	여러 페이지로 구성된 스냅샷(snapshot) 테이블에서, 현재 출력된 페이지 인덱스 숫자(1부터 시작).

원래 XData Contents 블록에서 <tablePane>을 통해 설정하는 *useSnapshot* 와 *pageSize* 속성 값은 프로그램을 통해서도 변경 가능하다.

#### ➤ 서버에서 데이터 가져오기

%ZEN.Component.tablePane 클래스는 스냅샷 테이블에서만 사용할 수 있는 *getRowData* 메소드를 제공한다. 이 메소드는 서버상의 스냅샷 데이터로부터 주어진 행(0부터 시작)의 데이터 값을 읽어낸다. 대상 데이터는 스냅샷 테이블의 컬럼명에 대응하는 프로퍼티들을 자바스크립트 오브젝트로 패키지화 한다. 스냅샷 모드가 아닌 테이블이나 범위 밖의 행에 대해서 *getRowData* 메소드는 Null 값을 반환한다.

#### ➤ 스냅샷 테이블 탐색

다중 페이지로 구성된 테이블의 페이지 탐색에는 여러 옵션들이 제공된다. <tableNavigator> 와 <tableNavigatorBar> 컴퍼넌트는 기본적인 탐색 인터페이스를 제공하며, 특히 <tableNavigatorBar>는 여러 페이지의 테이블을 관리하는데 유용하다(자세한 정보는 “내비게이션 버튼” 섹션을 참고).

프로그램을 통해 구현하는 경우, %ZEN.Component.tablePane 클래스는 응용프로그램으로 하여금 원하는 페이지 관리 인터페이스를 사용할 수 있도록 클라이언트 쪽 자바스크립트 API를 제공한다. 이러한 메소드들은 **tablePane**이 스냅샷 모드인 경우에만 적용되며, 다음과 같은 메소드들이 있다:

메소드	설명
<i>getPageCount()</i>	테이블의 총 페이지 수를 리턴.
<i>getProperty('currPage')</i>	현재 출력 페이지의 번호를 리턴.
<i>getProperty('pageSize')</i>	현재 페이지의 크기를 리턴.
<i>getProperty('rowCount')</i>	테이블의 전체 행수 리턴.  <i>rowCount</i> 타입은 숫자가 아닌 문자열로서 “” 또는 “100+”와 같은 형식을 취한다. 100 이상의 행은 “100+”로 표시한다. 자바스크립트에서 <i>rowCount</i> 값을 체크할 때, 숫자 형식으로 바꾸려면, 10 기반의 <i>parseInt</i> 함수를 사용한다:  <b>rowCount = parseInt(rowCount,10);</b>
<i>setProperty('currPage',pageno)</i>	현재 출력 페이지를 주어진 <i>pageno</i> 값으로 변경.
<i>setProperty('pageSize',rows)</i>	현재 출력 페이지 크기를 주어진 <i>rows</i> 값으로 변경.

## 컬럼 필터

Zen 테이블에서 컬럼의 헤더 위에 위치하는 “필터”를 생성할 수 있다. 필터는 하나 이상의 검색 조건을 입력할 수 있는 입력 상자를 말한다. 조건값이 입력되면, <tablePane>과 결합된 쿼리는 새 조건에 따라 재실행한 후, 페이지의 테이블 내용만을 갱신한다.

필터링은 <tablePane>에서 자동 생성 SQL 문 또는 *OnCreateResultSet* 콜백을 사용하는 경우에만 적용할 수 있는데 콜백은 데이터 필터링을 실행하기 위한 적절한 WHERE 논리를 생성한다. 테이블이 생성된 SQL 쿼리를 사용하면, 페이지 클래스는 입력값들을 %ZEN.Component.tablePane 프로퍼티인 *groupByClause*, *orderByClause*, *whereClause* 등에 형식화하여, 테이블 쿼리를 다시 실행한다.

**참고:** 필터를 설정하는 <column> 엘리먼트에 *colName* 값을 정의하지 않으면, Zen은 필터를 생성하지 않는다.

<column> 엘리먼트는 필터와 관련 다음과 같은 속성들을 제공한다.

속성	설명
<i>filterEnum</i>	<p><i>filterType</i> 값이 "enum" 이면, <i>filterEnum</i> 은 필터가 사용할 값들은 콤마 구분된 리스트로 정의한다. 예: "red,green,blue"</p> <p>정의된 값들은 콤보 상자에 표시되며, <i>filterEnumDisplay</i>가 정의되지 않는 한, <i>filterEnum</i> 리스트에 제공되는 이름들은 콤보 상자에 열거된다.</p>
<i>filterEnumDisplay</i>	<p><i>filterType</i> 값이 "enum" 이고, <i>filterEnumDisplay</i>가 콤마 구분 리스트 값을 제공하면, 콤보 상자는 대응하는 <i>filterEnum</i> 의 값 대신에 <i>filterEnumDisplay</i> 값을 표시한다.</p> <p><i>filterEnumDisplay</i> 속성은 1로 설정된 ZENLOCALIZE 데이터타입 파라미터를 갖는다. 이렇게 하면 텍스트를 현지 언어로 쉽게 바꿀 수 있으며, \$\$\$Text 매크로를 사용하여, 클라이언트 또는 서버 쪽 코드에서 이 프로퍼티로 값을 줄 수 있다.</p> <p>원래의 언어에서 콤마 구분 목록이었던 로컬 전환된 <i>filterEnumDisplay</i> 값은 모두 콤마 구분으로 구성되어야 한다.</p>
<i>filterLabel</i>	필터 콘트롤에 대한 레이블. ( <i>filterType</i> 값이 “range” 처럼) 복수의 필터 콘트롤이 존재하면, <i>filterLabel</i> 은 콤마로 구별된 복수의 레이블 리스트를 갖는다고 여긴다.
<i>filterOp</i>	해당 컬럼에 필터를 갖으면, <i>filterOp</i> 는 필터와 연관하여 사용되어지는 SQL 연산자명이다. 지원되는 연산자는 다음과 같다: "", "%STARTSWITH", "=", ">=", "<=", "<>", ">", "<", "[", "BETWEEN", "IN", "%CONTAINS".
<i>filterQuery</i>	<i>filterType</i> 값이 "query" 인 경우, <i>filterQuery</i> 는 드롭-다운 목록 값을 제공하는 SQL문을 정의한다. 쿼리가 하나 이상의 컬럼을 갖는 경우, 첫번째 컬럼은 (검색에 사용되는) 논리 값으로 사용하며, 두 번째 컬럼은 화면 표시값으로 사용된다.

<i>filterTitle</i>	사용자가 필터 콘트롤 위로 마우스를 움직일 때 나타나는 툴팁 텍스트. 이 속성값으로 일반적인 텍스트를 사용하지만 기본 데이터 타입은 %ZEN.Datatype.caption 이다. 이 값은 언어 DOMAIN 파라미터가 Zen 페이지 클래스에 정의되어 있으면 현지 언어로 쉽게 변경된다. 또한 %ZEN.Datatype.caption 데이터 타입은 \$\$\$Text 매크로를 사용하여 <i>cellTitle</i> 프로퍼티 값을 정의할 수 있도록 해 준다.
<i>filterType</i>	이 컬럼은 검색 필터 콘트롤을 나타내어야 한다는 것을 명시하며, 어떤 유형의 필터 콘트롤을 표시해야 하는지를 나타낸다. 사용 가능한 <i>filterType</i> 값으로는 "", "text", "range", "date", "datetime", "enum", "query", "custom" 등이 있다.
<i>filterValue</i>	현재 컬럼의 컬럼 필터 값. 일반적으로 이 속성은 사용자가 필터 콘트롤에 값을 입력한 후 값을 갖지만, <i>filterValue</i> 의 초기값을 정의하도록 설정할 수 있다. <i>filterValue</i> 는 다음과 같은 <i>filterOp</i> 값에 따라 의미를 달리 한다. <i>filterOp</i> 값이 <ul style="list-style-type: none"> <li>● "IN" 인 경우, <i>filterValue</i>는 IN 절에 콤마 구분 리스트 값으로 취급한다.</li> <li>● "%CONTAINS" 인 경우, <i>filterValue</i>는 %CONTAINS 절에 콤마 구분 리스트 값으로 취급한다.</li> <li>● "BETWEEN" 인 경우, <i>filterValue</i>는 쿼리의 BETWEEN 절에 사용되는 두 값의 콤마 구분 리스트 값으로 취급한다.</li> <li>● 그 외의 경우, <i>filterValue</i>는 단일 값으로 취급한다.</li> </ul>
<i>OnDrawFilter</i>	(옵션) Zen 페이지 클래스에서 서버 쪽 콜백 메소드명. 이 메소드는 HTML 콘텐츠를 &html<> 문법이나 WRITE 명령어를 사용하여 해당 컬럼의 필터로 삽입한다. 이 메소드는 컬럼을 작성할 때 호출되지만, 호출 시점에 <i>filterType</i> 의 값이 "custom"인 경우에만 호출된다. Zen은 메소드를 호출할 때, 아래의 파라미터들을 자동적으로 넘긴다. <ul style="list-style-type: none"> <li>● %ZEN.Component.tablePane : &lt;tablePane&gt; 오브젝트</li> <li>● %String : &lt;column&gt;에서의 <i>colName</i> 값</li> <li>● %ZEN.Auxiliary.column : &lt;column&gt; 오브젝트</li> </ul> 콜백은 %Status 데이터 타입을 반환해야 한다. 다음은 유효한 메소드 서명이다: <pre>Method DrawFil(pTable As %ZEN.Component.tablePane, pName As %String, pColinfo As %ZEN.Auxiliary.column) As %Status</pre> 위의 메소드를 콜백으로 사용하려면, <column>에서 OnDrawFilter="DrawFil"을 설정한다.

다음의 <tablePane> 예제에서는 employee의 Name과 Department를 나타내는 SQL (서술)문의 생성을 보여준다:

```

<tablePane id="table"
    useSnapshot="true"
    tableName="MyApp.Employee">
    <column colName="ID" hidden="true"/>
    <column colName="Name" filterType="text" />
    <column colName="Department"
        filterType="enum"
        filterEnum="Sales,Accounting,Marketing"
        filterOp="=" />
</tablePane>

```

위의 예제에서는, `<column> filterType` 속성을 사용하여, Name 컬럼이 컬럼 필터를 표시해야 한다는 것을 명시하고 있다 (“**text**”는 이 필터가 사용자가 텍스트를 입력할 수 있는 입력상자임을 표시한다). 사용자가 상자에 값을 입력하고(“A”와 같이) **Enter** 를 누르면, 테이블은 갱신되어 Name 컬럼에 “A”로 시작하는 행만을 보여준다.(`<column>`이 `filterOp` 값을 정의하지 않으면, 디폴트 매칭 작업은 **%STARTSWITH** 이 된다).

Department 컬럼에 대해서는 이 예제에서 3 개의 값을 보여주는 콤보 상자로 더욱 정교한 필터 기능을 표시하고 있다. 이를 위해서는, `<column> filterType` 를 “**enum**” 으로 설정하고, `filterEnum`을 가능한 값들의 콤마 구분 목록으로 설정한다. 또한 `filterOp` 값을 “**=**” 로 설정하여 정확한 매칭이 필요하다는 것을 표시한다.

`<tablePane>`에는 기본적으로 사용되는 필터가 있어, 필터 사용을 위해 별도로 `<tablePane>` 속성 값을 설정할 필요는 없다. 그러나 기본 설정을 변경할 필요가 있는 경우에는, `<tablePane>` 이 제공하는 다음의 속성들을 사용할 수 있는데, 이 속성들은 각각의 컬럼이 아닌 테이블 전체에 대한 필터링을 제어한다.

속성	설명
<code>autoExecute</code>	이 속성이 true 값을 가지면, 필터 값이 바뀔 때마다 테이블 쿼리가 실행된다. 디폴트 값은 true이며, false인 경우, 페이지는 <code>executeQuery</code> 메소드를 통해 쿼리가 실행되도록 <code>&lt;tablePane&gt;</code> 을 명시적으로 설정하여야 한다. 이 속성의(에는) 기본 데이터 타입은 <code>%ZEN.Datatype.boolean</code> 이며, 이 값은 XData Contents 블록에서는 true 또는 false, 서버 쪽 코드에서는 1 또는 0, 클라이언트에서는 true 또는 false로 가진다.
<code>filtersDisabled</code>	이 값이 true 면, 컬럼 필터가 수행되지 않는다. True인 경우, 컬럼 필터가 표시는 되지만 작동은 되지 않는다. 디폴트 값은 false이다. 이 속성의 기본 데이터 타입은 <code>%ZEN.Datatype.boolean</code> 이며, 이 값은 XData Contents 블록에서는 true 또는 false, 서버 쪽 코드에서는 1 또는 0, 클라이언트에서는 true 또는 false로 가진다.
<code>headerLayout</code>	컬럼 필터를 사용할 때, 테이블 헤더를 표시하는 방법을 콘트롤하며 다음의 값을 가질 수 있다:

	<ul style="list-style-type: none"> <li>● "filtersOnTop" : 컬럼 필터를 컬럼 헤더위로 위치시키는 것으로 디플트 값이다.</li> <li>● "headersOnTop" : 컬럼 헤더를 컬럼 필터위에 위치시킨다.</li> </ul>
<i>showFilters</i>	<p>이 값이 true 면, 컬럼 필터는 컬럼 헤더위에 위치한다. False 인 경우에는 필터가 표시되지 않는다. 디플트 값은 true .</p> <p>이 속성의 기본 데이터 타입은 %ZEN.Datatype.boolean 이며, 이 값은 XData Contents 블록에서는 true 또는 false, 서버 쪽 코드에서는 1 또는 0, 클라이언트에서는 true 또는 false로 가진다.</p>

### 컬럼 링크

Zen 테이블의 컬럼값으로 다른 페이지로 이동할 수 있는 페이지 링크를 정의할 수 있다 – 예를 들어, 해당 행의 오브젝트에 대한 보다 상세한 작업을 위한 오브젝트 편집 페이지로의 이동과 같은 경우. 이러한 링크는 데이터 값을 가지는 컬럼 자체 (이 경우, 데이터 값을 링크로 표시), 또는 링크를 가지는 별도의 컬럼으로 나타낼 수 있다.

<column> 엘리먼트는 링크와 관련, 다음과 같은 속성들로 제공한다.

속성	설명
<i>link</i>	<p>이 속성이 정의되면, 컬럼은 URI로서 <i>link</i> 프로퍼티 값을 표시한다. 링크에서 클라이언트의 자바스크립트 메소드를 호출하려면, 다음과 같이 자바스크립트으로 URI를 정의한다:</p> <pre>link="javascript:zenPage.myMethod();"</pre>
<i>linkCaption</i>	<p>이 컬럼이 링크를 정의하고 데이터 값을 표시하지 않으면, <i>linkCaption</i>은 링크의 제목을 정의한다.</p> <p>이 속성값으로 일반적인 텍스트를 사용하지만 기본 데이터 타입은 %ZEN.Datatype.caption 이다. 이 값은 언어 DOMAIN 파라미터가 Zen 페이지 클래스에 정의되어 있으면 현지 언어로 쉽게 변경된다. 또한 %ZEN.Datatype.caption 데이터 타입은 \$\$\$Text 매크로를 사용하여 <i>linkCaption</i> 프로퍼티 값을 정의할 수 있도록 해 준다.</p>
<i>linkConfirm</i>	<p>이 값이 설정되고 이 컬럼이 정의된 <i>link</i> 를 가지면, <i>linkConfirm</i> 텍스트는 링크를 실행하기 전의 확인 메시지로서 표시된다.</p> <p>이 속성값으로 일반적인 텍스트를 사용하지만 기본 데이터 타입은 %ZEN.Datatype.caption 이다. 이 값은 언어 DOMAIN 파라미터가 Zen 페이지 클래스에 정의되어 있으면 현지 언어로 쉽게 변경된다. 또한 %ZEN.Datatype.caption 데이터 타입은 \$\$\$Text 매크로를 사용하여 <i>linkConfirm</i> 프로퍼티 값을 정의할 수 있도록 해 준다.</p>

예제:

```
<tablePane id="table"
    sql="SELECT TOP 100 ID,Item,Price FROM MyApp.Inventory ORDER BY Price">
    <column colName="ID" hidden="true"/>
    <column colName="Item"
        link="MyApp.ItemEdit.cls?ID=%query.ID#"
        linkTitle="View information on this item."
    />
    <column link="javascript:zenPage.addToCart('%query.ID#');"
        linkCaption="Add to cart"
        linkConfirm="Do you wish to add this item to your cart?"
    />
</tablePane>
```

이 예제는 다음과 같은 작업을 수행한다:

1. <tablePane>의 *sql* 값은 MyApp.Inventory 테이블에서 상위 100 개 품목에 대한 정보를 가격순으로 표시하는 쿼리이다.
2. <tablePane>은 링크의 일부분으로 “ID” 컬럼 값이 필요하지만 이 값을 표시할 필요는 없기 때문에 <column>의 *hidden* 값은 true 이다.
3. <tablePane>은 특정 품목에 대한 정보를 표시하는 페이지에 링크를 갖기 위해 “Item” 컬럼이 필요하므로 <column> *link* 값은 이 링크를 정의한다. 또한 <column> *linkCaption* 값은 이 링크위로 마우스를 움직일 때 표시되는 툴팁 메시지를 정의한다.
4. <tablePane>은 데이터를 표시하지 않고 “Add to cart” 링크를 갖는 추가 컬럼을 정의하는데 이 링크를 클릭하면 클라이언트상의 **addToCart** 메소드가 호출된다. <column> *linkConfirm* 값을 지정하는 것은 사용자로 하여금 링크가 시행되기 전에 이 선택을 확인할 수 있는 기회를 주기 위함이다.

이 예제에 사용된 두 개의 링크는 Zen 표현식 문법을 사용하여, 링크에서 지정된 행의 데이터를 포함시키고 있다. 다음의 표현식은 테이블의 현재 행에 있는 ID 컬럼을 참조한다:

**#(%query.ID#)**

### 사용자 상호 작용

Zen 테이블은 테이블에 대한 페이지 검색, 컬럼 정렬, 행 선택등과 같은 기본적인 사용자 상호 작용을 지원하는 내장 메커니즘을 제공한다.

#### ➤ 내비게이션 버튼

<tableNavigator> 컴포넌트는 <tablePane> 페이지들간의 이동을 위한 버튼들을 자동적으로 제공한다. <tableNavigatorBar>는 동일한 방법으로 사용하지만 추가 버튼을 표시하여 큰 규모의 테이블을 내비게이트도 가능하도록 하고 있다.

이 컴포넌트를 사용하려면, <tablePane> 과 동일한 페이지의 임의의 위치에 컴포넌트를 정의하고 다음과 같이 *tablePanelId* 속성 값으로 <tablePane>의 *id* 값을 설정한다:

```
<pane id="tPane">
    <tablePane id="myTable"
        tableName="MyApp.Employee">
        <column colName="ID" hidden="true"/>
        <column colName="Name"/>
    </tablePane>
    <tableNavigatorBar tablePanelId="myTable" />
</pane>
```

<tableNavigator> 또는 <tableNavigatorBar>가 복합 엘리먼트안에 놓여진 경우, 대응하는 <tablePane>도 동일한 복합 엘리먼트내에 놓여져야 한다.

#### ➤ 내비게이션 (navigation) 키

<tablePane> 에서는 다음과 같이 사용자 키 클릭 이벤트를 정의할 수 있다.

프로퍼티	설명
<i>onkeypress</i>	<p><i>useKeys</i> 의 값이 true 면, 이 클라이언트상의 자바스크립트 표현식은 테이블의 키 (<b>Up</b>, <b>Down</b>, <b>Page Up</b>, <b>Page Down</b>, <b>Home</b>, <b>End</b>) 를 누를 때마다 실행한다.</p> <p>이 속성에 대한 값을 제공할 때에는, 큰 따옴표를 사용하여 값 전체를 묶고, 값 문자열 내부에 작은 따옴표를 사용하여 자바스크립트 표현식을 다음과 같이 구성한다:</p> <pre>&lt;tablePane onkeypress="alert('HEY');"/&gt;</pre> <p>자바스크립트 표현식에는 Zen의 #()# 실시간 표현식을 사용할 수 있다.</p>
<i>useKeys</i>	<p>이 값이 true (이)면, tablePane은 사용자 키스트로크(를 캡쳐) (<b>Up</b>, <b>Down</b>, <b>Page Up</b>, <b>Page Down</b>, <b>Home</b>, <b>End</b>) 를 캡쳐하여 간단한 테이블 내비게이션에 사용한다. 디폴트값(기본)은 false 이다.</p> <p>이 속성의(에는) 기본 데이터 타입은(형식인) %ZEN.Datatype.boolean 이( 있으)며, 이 값은 XData Contents 블록에서는 true 또는 false, 서버 쪽 코드에서는 1 또는 0, 클라이언트 코드(서버)에서는 true 또는 false로 가진다.</p>

#### ➤ 테이블 정렬

컬럼 헤더를 클릭하면, 이 컬럼의 데이터를 기준으로 테이블을 정렬한다. 첫번째 클릭은 오름차순으로, 그 다음 클릭은 내림차순으로 정렬되며 계속적인 클릭은 정렬 차순을 반복적으로 변경한다. <tablePane>을 페이지에 추가할 때, *sortOrder*에 대한 초기 값을 설정할 수도 있지만, 일반적으로 실행시간에 사용자에 의해 그 값이 설정된다.

프로퍼티	설명
<i>sortOrder</i>	<p>컬럼 헤더를 클릭하면, Zen은 테이블 쿼리를 재실행하고 (테이블이 스냅샷(snapshot) 모드가 아닌 경우) 해당 컬럼값과 <i>sortOrder</i>에 정의된 값을 기준으로 테이블을 정렬한다. 컬럼 헤더를 반복적으로 클릭함으로서 “asc”(오름차순)과 “desc”(내림차순) 값이 반복적으로 설정된다.</p> <p><i>sortOrder</i>는 쿼리 자체에는 영향을 주지 않으며 (쿼리의 ORDER BY 설정과 관계 없음), 단지 쿼리의 결과로 테이블에 리턴된 resultset의 순서를 제어할 뿐이다.</p> <p><i>sortOrder</i> 값은 문자열이거나, Zen의 #()# 실시간 표현식으로 정의 가능하다.</p>

#### ▶ 행과 컬럼의 선택

Zen 테이블에서 원하는 행과 컬럼을 선택할 수 있다. 현재 선택된 행은 알아보기 쉽게 표시되고 <tablePane> *onselectrow* 속성에서 제공되는 이벤트 핸들러를 실행시켜 해당 이벤트의 어플리케이션에 알려준다.

%ZEN.Component.tablePane 은 행과 컬럼 선택과 관련된 프로퍼티들을 지원하고 있는데 이 중 많은 프로퍼티들이 <tablePane> 속성으로서 설정가능한 반면, 몇몇 프로퍼티들은 실행 시에만 사용자에 의해 값이 결정된다.

프로퍼티	설명
<i>currColumn</i>	가장 마지막에 선택된 컬럼의 <i>colName</i> . 이 프로퍼티의 값은 사용자 액션 또는 직접적인 설정으로 제공할 수 있다. <i>currColumn</i> 값은 문자열이거나, Zen의 #()# 실시간 표현식으로 정의 가능하다.
<i>multiSelect</i>	이 값이 true 면, 복수의 테이블 행을 선택할 수 있다. Zen은 체크 박스를 갖는 여분의 추가 컬럼을 표시하여 선택된 행을 나타낸다. 디폴트 값은 false이다. <i>multiSelect</i> 와 <i>rowSelect</i> 는 서로 독립적으로 true 또는 false 값을 설정할 수 있다.
<i>ondblclick</i>	<p>현재 테이블의 특정 행을 더블 클릭할 때마다 실행되는 클라이언트상의 자바스크립트 표현식. 일반적으로 이 표현식은 페이지 클래스에서 정의한 클라이언트의 자바스크립트 메소드를 호출한다. 이 메소드는 &lt;tablePane&gt; 에 대한 “<i>ondblclick</i> 이벤트 핸들러” 가 된다.</p> <p><i>ondblclick</i> 과 같은 이벤트 핸들러 속성에 대한 값을 제공할 때, 다음 예제와 같이, 큰 따옴표를 사용하여 자바스크립트내의 값과 작은 따옴표를 묶는다:</p> <pre>&lt;tablePane ondblclick="alert('HEY');"/&gt;</pre> <p>자바스크립트는 Zen의 #()# 실시간 표현식으로 정의 가능하다.</p>
<i>onheaderClick</i>	테이블의 특정 컬럼 헤더를 클릭할 때마다 실행되는 클라이언트상의 자바스크립트 표현식. Zen은 이 컬럼명을 <i>currColumn</i> 프로퍼티에 저장한다.

<i>onselectrow</i>	<i>rowSelect</i> 값이 true 면, 테이블에서 새로운 행을 선택할 때마다 실행되는 클라이언트 자바스크립트 표현식. <i>showRowSelector</i> 가 true 인 경우에만 적용된다.
<i>onmultiselect</i>	<i>multiSelect</i> 값이 true 면, 테이블에서 선택된 복수의 행들의 선택을 변경할 때마다 실행되는 클라이언트 자바스크립트 표현식.
<i>rowSelect</i>	이 값이 true 면, 한 번에 하나의 행을 선택할 수 있다. 디폴트 값은 true이며 <i>multiSelect</i> 와 <i>rowSelect</i> 는 서로 독립적으로 true 또는 false 값을 가질 수 있다.
<i>selectedIndex</i>	(0-기반의) 현재 선택된 행의 인덱스 번호. <i>showRowSelector</i> 프로퍼티 값이 true 인 경우에만 이 값이 적용된다. 스냅샷(snapshot) 테이블의 경우, 이 값은 현재 페이지에서의 행 번호를 의미한다.
<i>selectedRows</i>	읽기 전용. <tablePane>에서 <i>multiSelect</i> 값이 true이고 <i>valueColumn</i> 값이 정의되어 있는 경우, <i>selectedRow</i> 의 문자열 값은 현재 선택된 행들을 나타낸다. 이 값은 선택된 각 행의 <i>valueColumn</i> 값을 콤마 구별 목록으로 제공한다. 값이 포함되지 않은 연속 콤마의 경우, 해당 위치의 행은 선택되지 않았다는 것을 의미한다.  예를 들어, 다음과 같이 <i>selectedRow</i> 값이 정의되었다면, 14 개의 행 가운데 3 개의 행만 선택되었다는 의미가 된다:  “...value,,,value,,,,,value,,”  만약 <tablePane> 이 정의된 <i>valueColumn</i> 를 갖고 있지 않은 상태에서 <i>multiSelect</i> 값이 true 로 설정되었다면, <i>selectedRow</i> 값은 다음과 같은 문자열을 제공하게 된다:  “,,,,,,,,,,,”
<i>showRowSelector</i>	이 값이 true 이고 선택된 행들이 존재하면, 테이블은 선택된 행을 나타내는 추가의 컬럼을 표시한다. 디폴트 값은 true 이다.
<i>value</i>	읽기 전용. 이 값은 현재 어떤 행이 선택되어 있는지를 결정하는데 사용되는 논리적 값이다.  <i>Value</i> 는 <i>valueColumn</i> 과 함께 적용되며, 값으로 빈 문자열("")을 가질 수도 있다.  이 속성값을 직접 액세스하지는 않으며, <b>getProperty('value')</b> 을 사용하여 값을 얻는다.
<i>valueColumn</i>	<tablePane>은 논리적 값을 가질 수 있는데, 테이블의 새로고침이 실행될 때마다, Zen은 각 행의 이 논리적 값을 해당 행의 <i>valueColumn</i> 의 실제 값과 비교하여 <i>valueColumn</i> 에 값을 갖고 있는 모든 행을 선택하는데, 이것은 다음과 같은 의미를 가진다:

- <tablePane>의 값과 테이블이 처음 표시될 때 선택되는 행들을 사전에 미리 설정할 수 있다.
- 테이블의 행을 정렬할 때, 현재의 선택이 그대로 유지된다.

## 테이블 새로 고침

테이블을 새로 고치면, 테이블만이 새로 고쳐진다. 즉, 테이블을 새로 고치기 위해 전체 Zen 페이지가 새로 고치질 필요는 없다.

일반적으로, Zen은 필요에 따라 테이블의 출력된 내용을 자동적으로 새로 고친다. 프로그램에서 테이블을 정의하는 경우에는 다음과 같은 방법을 사용하여 명시적으로 테이블을 새로 고칠 수 있다:

- %ZEN.Component.tablePane의 *refreshRequired* 프로퍼티를 서버 메소드에서 **true(1)**로 설정한다. 이 값은 **tablePane**으로 하여금 쿼리를 재 실행하도록 한다.
- %ZEN.Component.tablePane의 *executueQuery* 메소드를 호출하면 데이터 쿼리가 재실행되고 테이블의 값들이 업데이트되어 **tablePane** 프로퍼티의 현재 값을 반영한다.

## 테이블 수정(touchup)

XData Contents 블록에서 <tablePane>의 속성 값을 설정하게 되면, **tablePane** 오브젝트의 대응하는 프로퍼티는 자동적으로 이 값을 갖게 되는데 이러한 기능만으로도 원하는 프로그램 구성은 충분히 할 수 있지만, %ZEN.Component.tablePane는 클라이언트 또는 서버 상의 프로그램적 상호 작용을 위한 같은 추가 기능들을 제공한다.

### ➤ 데이터 값

테이블에 표시되는 데이터 값을 수정하려면 다음과 같은 옵션을 사용한다.

- 테이블을 출력하기 전에 페이지 클래스의 %OnAfterCreatePage 콜백의 **tablePane** 프로퍼티들을 설정하여, XData Contents 블록에서 설정된 값을 수정한다.
- 테이블의 현재 상태를 결정하는 *lastUpdate* 와 *rowCount* 같은 %ZEN.Component.tablePane의 읽기 전용 프로퍼티들을 체크한다. *rowCount*의 값은 "" 또는 "100+"처럼 숫자가 아닌 문자열이라는 점을 유의하여야 한다. 100 개 이상의 행은 "100+"로 나타낸다.
- **tablePane**의 특정 프로퍼티 값을 다시 설정하고, 테이블을 새로 고친다.

### ➤ 헤더와 본문 정렬

Internet Explorer 를 사용할 때 *fixedHeaders* 값을 **true**로 설정하게 되면, <tablerPane> 헤더와 본문 컬럼들은 정렬이 잘 되지 않는다. 이러한 이유로, %ZEN.Component.tablePane 클래스는 *resizeHeaders*라는 클라이언트에서 실행되는 자바스크립트 메소드를 제공하여 주어진 테이블의

정렬 상태를 확인하고, 필요한 경우, 테이블 본문에 수직 스크롤 바가 차지하는 공간을 확보하기 위해 헤더를 재조정한다.

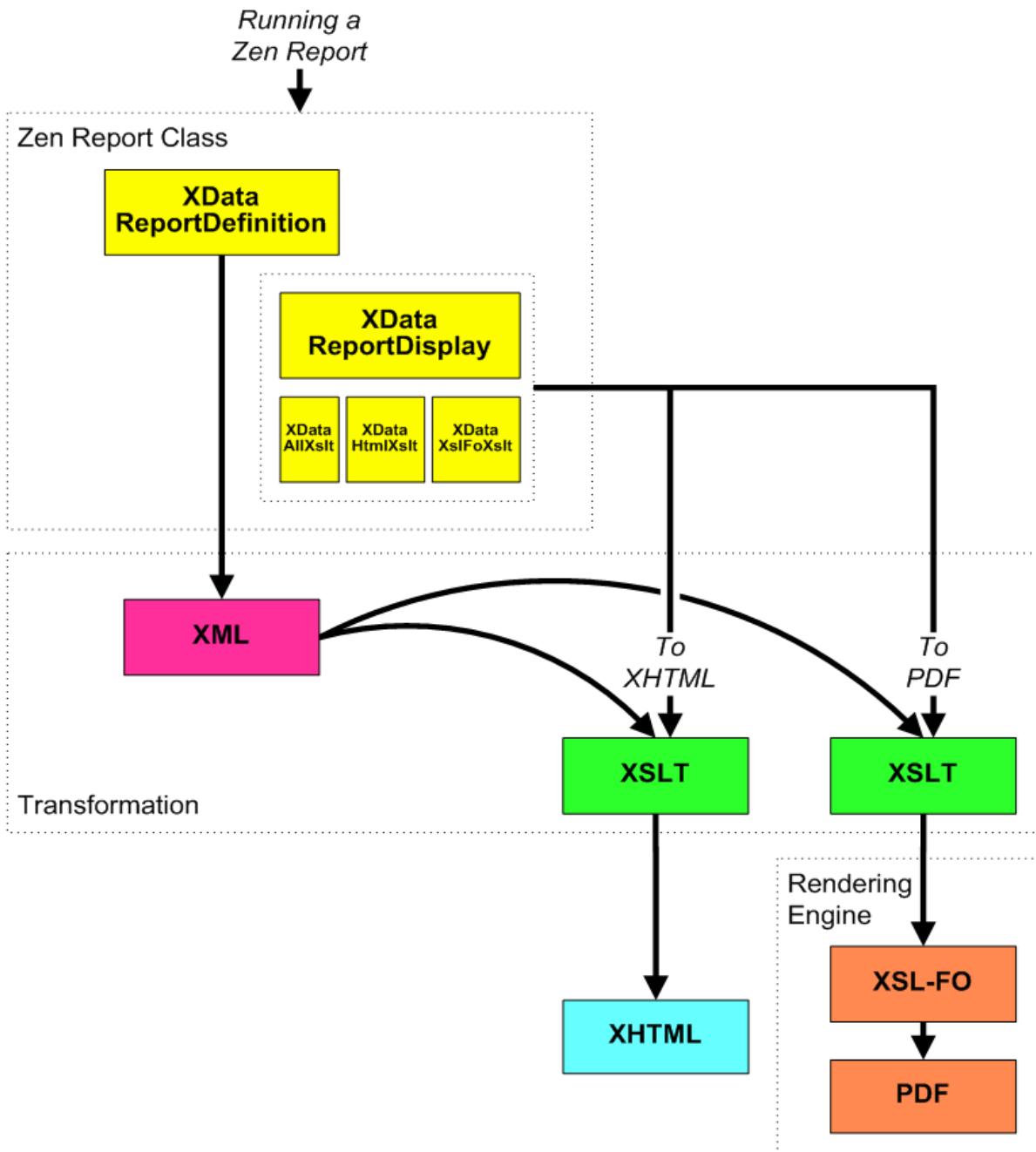
*resizeHeaders*는 출력된 페이지의 스크롤 바의 실제 크기를 바탕으로 여백의 크기를 계산하여 자동적으로 Internet Explorer 6 과 7 간의 차이 뿐만 아니라 윈도우즈 데스크탑 테마 부분(theme)에서의 스크롤바 넓이 문제로 발생하는 사소한 차이를 해소해 준다. *fixedHeaders* 값을 true로 설정하고 <tablePane> 컴포넌트를 구성할 때, Internet Explorer에 문제가 없는 한, *resizeHeaders*를 사용할 필요는 없다. 만약 문제가 발생한다면 다음과 같이 해결할 수 있다:

1. <tablePane>을 출력하는 Zen 페이지 클래스의 클라이언트 쪽 자바스크립트 메소드 *onresizeHandler*를 실행한다. *onresizeHandler*는 페이지 클래스에서 실행되어야 하는데, 이는 resize 이벤트만이 현재의 zenPage 오브젝트에 대하여 지원되고 페이지의 다른 컴포넌트들에는 영향을 주지 않기 때문이다.
2. *onresizeHandler* 실행시, **tablePane** 오브젝트의 *resizeHeaders* 메소드를 명시적으로 호출해야 한다.

## 2) Zen report 만들기

Zen은 Zen Report라는 Caché 데이터베이스를 기반으로 한 XHTML 또는 PDF 형식의 리포트를 작성하는 확장형 프레임워크를 제공한다. Zen은 다음과 같이 세 가지 독립된 절차에 따라 리포트를 작성한다:

1. Zen 리포트 클래스를 생성한 후, 이 클래스에서 다음의 두 XML 문서를 제공한다:
  - XData ReportDefinition을 사용하여 데이터 정의
  - XData ReportDisplay를 사용하여 출력 형식 정의
2. 브라우저에서 리포트 클래스 URI를 사용하거나 어플리케이션에서 **GenetateReport** 메소드를 호출하여 리포트를 출력하는데 출력 타입은 다음과 같아야 한다:
  - 브라우저를 사용하여 XHTML 형식의 리포트 출력
  - PDF 형식의 프린터 출력 페이지 생성
3. 리포트 클래스는 아래와 같은 절차를 통해 데이터 및 출력형식 정보를 전달한다:



두 가지 형식의 Zen 리포트의 주요 프로세싱은 다음과 같다:

#### To XHTML

XData ReportDefinition01 쿼리를 실행하면 Zen은 쿼리에서 리턴된 결과를 XML 형식으로 구성한다. Xdata ReportDisplay는 **To-HTML stylesheet**라는 XSLT 변환을 생성하고 Zen은 이 변환을 XML 파일에 적용하여 XHTML 출력 파일이 구성된다.

#### To PDF

XData ReportDefinition01 쿼리를 실행하면 Zen은 쿼리에서 리턴된 결과를 XML 형식으로 구성한다. Xdata ReportDisplay는 **To-XSLFO stylesheet**라는 XSLT 변환을 생성하고 Zen은 이 변환과 XML 파일을 XSL-FO 형식의 리포트를 생성하는 제 3의 랜더링 툴에 전송한다. 그러면 해당 툴은 이 변환과 XML 파일을 PDF 형식의 XSL-FO로 랜더링한다.

다음은 PDF 형식의 리포트 출력 예이다.

Patient History							6/7
20050315	12:36:10	TRO		97 %			
	BANACH S	AC Inform	334934		334934		
	Pu						Active
<u>Site:</u>	MIT		<u>Sending Unit:</u>				
<u>Last Name:</u>	Olsen		<u>First Name:</u>	Heloisa			
<u>Patient ID:</u>	48915		<u>Date of Birth:</u>	09/19/64			
<u>Sex:</u>	M		<u>Location:</u>				
Date	Time	Test	Result Value	Unit	Result Limits	Test LOT	
	Operator	Inst. Type	Instr. Name		Instr. S/N		Valid.Status
20060124	04:26:26	MIK		46 mmHg			
	EINSTEIN A	OMNI C	415372		415372		
	I						Active
20050813	19:57:52	TRO		67 L			
	BANACH S	OMNI C	245403		245403		
	Ism						Active
20041126	07:17:31	JIM		94 mmol/L			
	HAWKING S	OMNI C	470239		470239		
	M						Inactive
20040604	10:17:03	KEI		66 %			
	HAWKING S	OMNI C	331692		331692		
	Zen						Inactive
Report Generation Demonstration **SAMPLE**							

## Zen report 클래스 생성

Zen 리포트는 %ZEN.Report.reportPage를 확장한 클래스이다. 이 섹션에서는 Zen 리포트 클래스 정의 절차를 차례대로 구현함으로써 Zen 리포트의 구조에 대한 이해를 설명하고자 한다.

이 섹션을 시작하기 전에, 설치된 Caché 시스템의 SAMPLES 네임스페이스에서 필요한 데이터를 먼저 생성하여야 한다. 브라우저에서 다음 URI를 입력하면 설치시 제공되는 Zen 샘플들을 볼 수 있다:

<http://localhost:57772/csp/samples/ZENDemo.Home.cls>

여기에서, 57772는 Caché 설치시 지정된 Caché 웹 서버 포트 번호이다.

이제 Caché Studio 를 통해 Zen 리포트 정의를 시작한다:

1. Caché Studio를 시작한다.
2. 파일 > 네임스페이스 변경 또는 F4를 선택한다.
3. SAMPLES 네임스페이스를 선택한다.
4. 파일 > 새로 만들기 또는 Ctrl-N를 선택한다.

5. “새로 만들기” 창에서 Zen 탭을 선택한다.
6. ‘새 Zen 리포트’ 아이콘을 클릭한다.
7. ‘확인’ 버튼을 클릭한다.

Zen 리포트 마법사는(Zen Report Wizard) 다음의 필드들에 대한 입력란을 제공한다. 오른쪽의 입력값 컬럼은 이 섹션에서 설명과 더불어 생성하는 클래스의 입력값이다.

필드	내용	입력하는 값
패키지명	리포트 클래스를 포함할 패키지	MyApp
클래스명	리포트 클래스명	ReportDemo
애플리케이션	이 리포트가 속하는 애플리케이션의 패키지와 클래스명	이 필드값은 리포트 생성에는 관련없는 문서를 위한 값으로서 설정하지 않아도 된다.
리포트명	애플리케이션에서 사용할 리포트의 논리적 이름	MyReport
설명	리포트에 대한 설명	리포트 생성 샘플

‘다음’ 버튼을 클릭한다.

8. SQL 쿼리 입력란에 다음의 쿼리를 입력하여 리포트 데이터를 생성한다:

```
SELECT ID,Customer,Num,SalesRep,SaleDate
FROM ZENApp_Report.Invoice ORDER BY SalesRep,Customer
```

‘마침’ 버튼을 클릭한다.

리포트 마법사는 마법사를 통해 미리 정의한 파라미터들과 XML 블록인 XData ReportDefinition 및 XData ReportDisplay로 구성된 Zen 리포트 페이지 클래스를 자동 생성한다.

9. XData ReportDefinition 블록에서 아래의 위치를 찾아, 이 주석 라인과 </report> 엘리먼트 사이에 커서를 위치 시킨다.

```
<!-- add definition of the report here. -->
```

10. 리포트는 하나 이상의 그룹으로 구성되는데 XData ReportDefinition 내에 </report> 앞에 놓은 커서 위치에 다음과 같이 <group> 엘리먼트를 추가하여 첫번째 그룹을 정의한다:

```
<report xmlns="http://www.intersystems.com/zen/report/definition"
    name="MyReport"
    sql="SELECT ID,Customer,Num,SalesRep,SaleDate
        FROM ZENApp_Report.Invoice ORDER BY SalesRep,Customer" >
    <group name="SalesRep" breakOnField="SalesRep">
    </group>
</report>
```

11. 빌드 > 컴파일, 또는 Ctrl-F7 을 선택한다.
12. ‘보기 > 웹 페이지’를 선택하여 브라우저를 통해 생성된 리포트 페이지를 출력한다. Studio에서 바로 브라우저를 호출하여 Zen 리포트 페이지를 보는데 문제가 있으면, 브라우저를 시작한 후 다음과 같이 클래스명을 입력한다.

<http://localhost:57772/csp/samples/MyApp.ReportDemo.cls>

여기에서, 57772 은 Caché 의 웹 서버의 포트 번호이다.

브라우저에 출력되는 리포트 데이터의 XML 뷰는 다음과 같은데 일단 데이터는 없이 정의된 구조만 보인다:

```
<?xml version="1.0" ?>
<ReportDemo>
  <SalesRep></SalesRep>
  <SalesRep></SalesRep>
  <SalesRep></SalesRep>
  <SalesRep></SalesRep>
  <SalesRep></SalesRep>
  <SalesRep></SalesRep>
</ReportDemo>
```

13. <attribute> 엘리먼트를 XData ReportDefinition 블록의 <group>에 추가한다:

```
<report xmlns="http://www.intersystems.com/zen/report/definition"
  name="MyReport"
  sql="SELECT ID,Customer,Num,SalesRep,SaleDate
    FROM ZENApp_Report.Invoice ORDER BY SalesRep,Customer" >
  <group name="SalesRep" breakOnField="SalesRep">
    <attribute name="name" field="SalesRep" />
  </group>
</report>
```

14. 클래스를 컴파일하고 브라우저에서 리포트를 확인한다. 이번에는 각 <SalesRep> 엘리먼트가 *name* 속성을 갖고 있기 때문에, 이 속성의 데이터가 SQL 쿼리로 반환되어 출력된다:

```
<?xml version="1.0" ?>
<ReportDemo>
  <SalesRep name="Jack"></SalesRep>
  <SalesRep name="Jen"></SalesRep>
  <SalesRep name="Jill"></SalesRep>
  <SalesRep name="Jim"></SalesRep>
  <SalesRep name="Joanne"></SalesRep>
  <SalesRep name="John"></SalesRep>
</ReportDemo>
```

15. <aggregate> 엘리먼트를 XData ReportDefinition 블록내의 group에 추가한다:

```
<report xmlns="http://www.intersystems.com/zen/report/definition"
  name="MyReport"
  sql="SELECT ID,Customer,Num,SalesRep,SaleDate
    FROM ZENApp_Report.Invoice ORDER BY SalesRep,Customer" >
```

```

<group name="SalesRep" breakOnField="SalesRep">
    <attribute name="name" field="SalesRep" />
    <aggregate name="total" type="SUM" field="Num" />
</group>
</report>

```

16. 클래스를 컴파일하고 다시 브라우저를 통해 리포트를 확인한다. 각 <SalesRep> 엘리먼트는 <total>이라는 엘리먼트를 추가하였고 <total>의 값은 SQL 쿼리에서 반환되는 SalesRep 필드의 모든 Num 필드 값의 총계를 나타낸다.

```

<?xml version="1.0" ?>
<ReportDemo>
    <SalesRep name="Jack">
        <total>833</total>
    </SalesRep>
    <SalesRep name="Jen">
        <total>774</total>
    </SalesRep>
    <SalesRep name="Jill">
        <total>983</total>
    </SalesRep>
    <SalesRep name="Jim">
        <total>826</total>
    </SalesRep>
    <SalesRep name="Joanne">
        <total>824</total>
    </SalesRep>
    <SalesRep name="John">
        <total>825</total>
    </SalesRep>
</ReportDemo>

```

17. XData ReportDefinition 내에 <aggregate>, <element>, 및 <group> 엘리먼트를 추가한다.

```

<report xmlns="http://www.intersystems.com/zen/report/definition"
    name="MyReport"
    sql="SELECT ID,Customer,Num,SalesRep,SaleDate
        FROM ZENApp_Report.Invoice ORDER BY SalesRep,Customer" >
    <group name="SalesRep" breakOnField="SalesRep">
        <attribute name="name" field="SalesRep" />
        <aggregate name="total" type="SUM" field="Num" />
        <aggregate name="average" type="AVG" field="Num" />
        <aggregate name="clients" type="COUNT" field="Customer" />
        <group name="SalesTo" breakOnField="Customer" >
            <element name="customer" field="Customer" />
            <attribute name="date" field="SaleDate" />
        </group>
    </group>
</report>

```

18. 클래스를 컴파일하고 리포트를 확인한다. XML 형식의 출력 데이터는 각 영업 담당자의 전체 데이터이다. 집합적 엘리먼트 <total>, <client>, 와 <average> 등이 각 <SalesRep> 레코드의 끝에 위치한다.

19. 이제 XML 형식으로 구축된 리포트 데이터를 XHTML 또는 PDF 형식으로 출력하는 방법을 정의한다.

XData ReportDisplay 블록의 아래 텍스트를 찾아 이 라인과 </report> 엘리먼트 사이에 커서를 위치시킨다.

```
<!-- add display definition of the report here. -->
```

20. 커서 위치에서 아래와 같이 <body> 엘리먼트를 추가한다. <body>내에 위치한 <p>는 리포트의 제목을 출력한다. 또한 <report> 엘리먼트에 *title* 속성을 추가하여 브라우저 윈도우의 제목을 설정한다:

```
<report xmlns="http://www.intersystems.com/zen/report/display"  
        name="MyReport" title="Tutorial Sales Report">  
    <body>  
        <p>Tutorial Sales Report</p>  
    </body>  
</report>
```

21. DEFAULTMODE 클래스 파라미터 값을 “xml”에서 “html”로 바꾼다.

22. 클래스를 컴파일하고 리포트를 확인한다.

23. 테이블을 XData ReportDisplay에 다음과 같이 추가한다.

```
<report xmlns="http://www.intersystems.com/zen/report/display"  
        name="MyReport" title="Tutorial Sales Report">  
    <body>  
        <p>Tutorial Sales Report</p>  
        <group name="SalesRep" line="1px">  
            <table orient="row" width="4in">  
                <item field="@name" width="2in">  
                    <caption value="Sales Rep:" width="2in"/>  
                </item>  
                <item field="total" formatNumber="#0.00">  
                    <caption value="Total Value of Sales:"/>  
                </item>  
                <item field="clients">  
                    <caption value="Number of Clients:"/>  
                </item>  
            </table>  
        </group>  
    </body>  
</report>
```

여기에서

- <group name="SalesRep">는 생성된 XML의 <SalesRep> 엘리먼트에 대한 참조이다.
- <item field="@name">는 <SalesRep> 속성 *name* 을 참조하는 문법이다.
- <item field="total">은 <SalesRep> 엘리먼트 <total>을 참조하는 문법이다

24. 클래스를 컴파일하고 리포트를 확인한다.

25. 사전 정의된 style 클래스를 사용하여 제목을 강조하기 위해 <p> 엘리먼트를 다음과 같이 변경한다:

```
<p class="banner1">Tutorial Sales Report</p>
```

26. 클래스를 컴파일하고 리포트를 확인한다.

27. XData ReportDisplay 블록을 다음과 같이 추가로 변경하여 페이지의 스타일과 배치를 수정하고 출력될 데이터도 추가한다:

```
<report xmlns="http://www.intersystems.com/zen/report/display"
        name="MyReport" title="Tutorial Sales Report">
    <body>
        <p class="banner1">Tutorial Sales Report</p>
        <group name="SalesRep" line="1px">
            <line pattern="empty"/>
            <table orient="row" width="4in">
                <item field="@name" width="2in">
                    <caption value="Sales Rep:" width="2in"/>
                </item>
                <item field="total" formatNumber="##0.00">
                    <caption value="Total Value of Sales:"/>
                </item>
                <item field="average" formatNumber="##0.00">
                    <caption value="Average Individual Sale:"/>
                </item>
                <item field="clients">
                    <caption value="Number of Clients:"/>
                </item>
            </table>
            <line pattern="empty"/>
            <table orient="col" group="SalesTo" altcolor="#FFDFDF" width="3.8in">
                <item field="customer" >
                    <caption value="Customers:"/>
                </item>
                <item field="@date" >
                    <caption value="Date of Sale:"/>
                </item>
            </table>
        </group>
    </body>
</report>
```

28. 클래스를 컴파일하고 리포트를 확인한다.

## 웹 브라우저에서의 Zen Report

브라우저에서 Zen 리포트 .cls 파일을 URI 값으로 입력하면 XHTML 과 PDF 리포트를 볼 수 있다. 출력 형식을 결정하기 위해서는, Zen 리포트 클래스의 DEFAULTMODE 클래스 파라미터를 사용하거나, URI 값에 \$MODE 파라미터를 제공한다. 다음은 \$MODE를 사용하는 예제이다.

- HTML 리포트를 보는 경우

[http://localhost:57772/csp/myPath/myApp.myReport.cls?\\$MODE=html](http://localhost:57772/csp/myPath/myApp.myReport.cls?$MODE=html)

여기에서, 57772 은 Caché 서버에 지정된 웹 서버 포트 번호이다.

**참고:** 사용자 환경 설정에 있는 포트 번호는 다를 수 있으므로, 시스템 관리 포탈의 [홈] > [환경 구성] > [시작 설정] 페이지의 “WebServerPort”에서 설정된 정확한 포트 번호를 확인할 수 있다.

HTML 형식의 리포트는 다음과 같다:

The screenshot shows a Microsoft Internet Explorer window displaying a sales report titled "HelpDesk Sales Report". The report header includes the title, month (Jul), author (BOB), prepared by (UnknownUser), and time (2007-04-12 21:39:51). Below the header is a table with three rows: Sales Rep (Jack), Number of Sales (22), and Total Value of Sales (121.00). The main content is a table titled "Average Sale: 5.500" showing 22 sales entries with columns for #, Sale ID, Date, Customer, and Amount. The total amount for all sales is 121.00. The browser interface at the top includes the address bar with the URL http://127.0.0.1:57772/csp/samples/ZENApp.MyReport.cls?MONTH=7&\$MODE=html, and various menu items like File, Edit, View, Favorites, Tools, Help, and a toolbar with links to Custom Links, Free Hotmail, Windows Marketplace, and Windows Media.

Sales Rep:	Jack
Number of Sales:	22
Total Value of Sales:	121.00

Average Sale: 5.500				
#	Sale ID	Date	Customer	Amount
1	472	2005-07-01	IntraSonics Partners	8.00
2	664	2005-07-01	XenaMatix Corp.	8.00
3	828	2005-07-01	CompuWare Media Inc.	9.00
4	425	2005-07-02	AccuPlex Associates	9.00
5	139	2005-07-06	SecuriDynamics GmbH.	2.00
6	679	2005-07-06	YoyoTron LLC.	5.00
7	178	2005-07-07	MegaData LLC.	8.00
8	549	2005-07-08	DynaSys Associates	1.00
9	296	2005-07-09	XenaGlomerate LLC.	4.00
10	295	2005-07-11	PicoPlex Corp.	9.00
11	913	2005-07-14	MediSoft Holdings Inc.	10.00
12	867	2005-07-15	PicoGlomerate Associates	1.00
13	420	2005-07-16	TeraPedia Group Ltd.	8.00
14	77	2005-07-17	DynaSoft Corp.	6.00
15	306	2005-07-17	Picomo Partners	4.00
16	257	2005-07-20	YoyoPlex Group Ltd.	6.00
17	502	2005-07-21	AccuDynamics LLC.	3.00
18	525	2005-07-21	CyberGlomerate Inc.	9.00
19	620	2005-07-24	Secuniry LLC.	2.00
20	1	2005-07-26	RoboSystems.com	8.00
21	567	2005-07-30	InterSys Group Ltd.	0.00
22	975	2005-07-31	TeleCalc Inc.	1.00
				121.00

- PDF 리포트를 보기 위해서는, 우선 “PDF 출력을 위한 Zen 구성” 섹션의 절차에 따라 환경 구성을 한 다음 아래와 같이 URI를 제공한다:

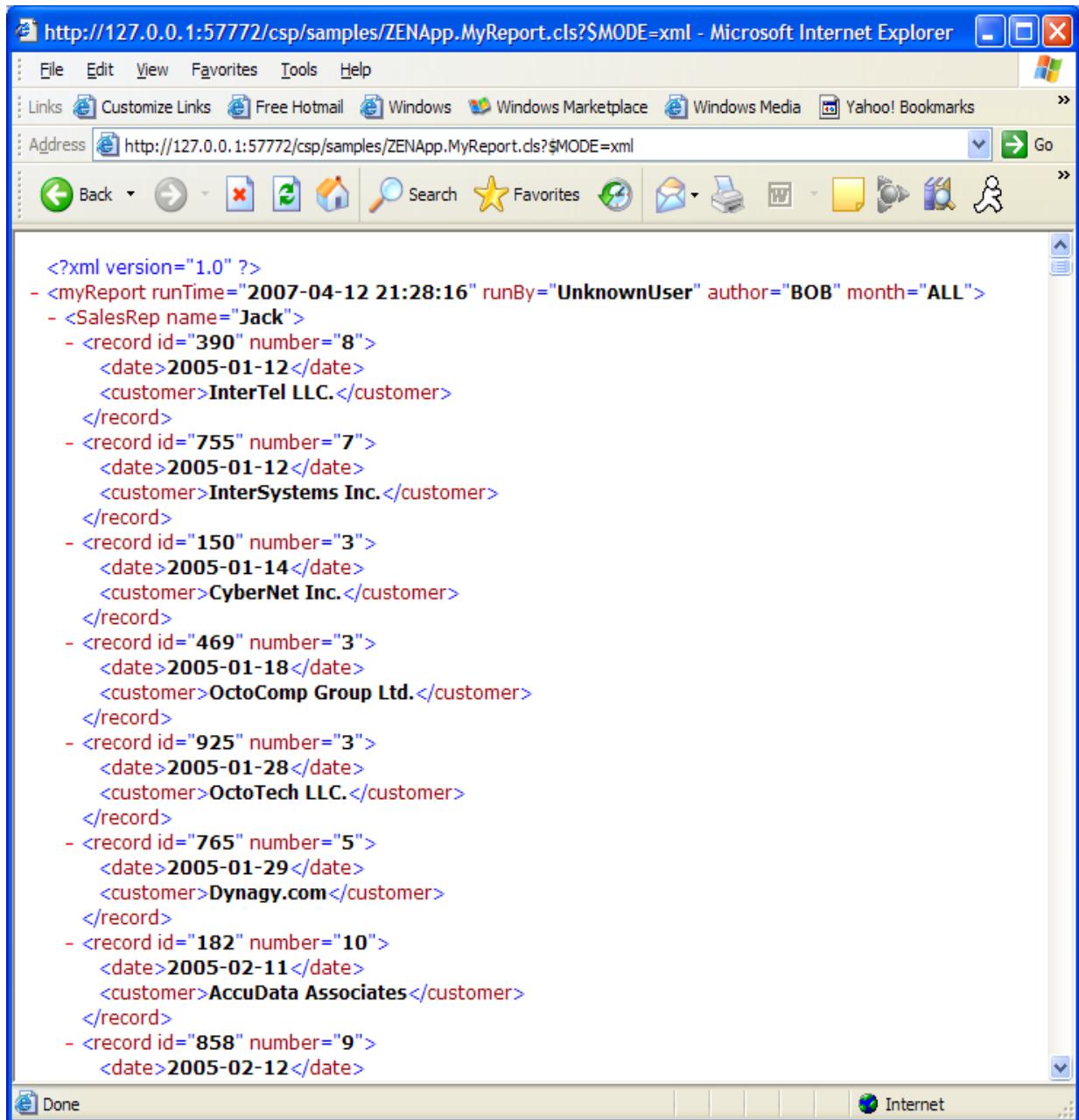
`http://localhost:57772/csp/myPath/myApp.myReport.cls?$MODE=pdf`

그러면 리포트는 PDF 형식으로 나타나며, 사용자 설정에 따라, 파일 저장 여부를 브라우저가 물어볼 수도 있다. 그런 경우, PDF를 보기 위해 ‘저장’을 클릭한다.

- XML 형식으로 리포트의 원자료(raw data)를 보기 위해서는 다음과 같이 URI를 입력한다.

`http://localhost:57772/csp/myPath/myApp.myReport.cls?$MODE=xml`

그러면 다음과 같이 XML 형식의 데이터가 출력된다.



The screenshot shows a Microsoft Internet Explorer window displaying an XML document. The title bar reads "http://127.0.0.1:57772/csp/samples/ZENApp.MyReport.cls?\$MODE=xml - Microsoft Internet Explorer". The XML content is as follows:

```

<?xml version="1.0" ?>
- <myReport runTime="2007-04-12 21:28:16" runBy="UnknownUser" author="BOB" month="ALL">
  - <SalesRep name="Jack">
    - <record id="390" number="8">
      <date>2005-01-12</date>
      <customer>InterTel LLC.</customer>
    </record>
    - <record id="755" number="7">
      <date>2005-01-12</date>
      <customer>InterSystems Inc.</customer>
    </record>
    - <record id="150" number="3">
      <date>2005-01-14</date>
      <customer>CyberNet Inc.</customer>
    </record>
    - <record id="469" number="3">
      <date>2005-01-18</date>
      <customer>OctoComp Group Ltd.</customer>
    </record>
    - <record id="925" number="3">
      <date>2005-01-28</date>
      <customer>OctoTech LLC.</customer>
    </record>
    - <record id="765" number="5">
      <date>2005-01-29</date>
      <customer>Dynagy.com</customer>
    </record>
    - <record id="182" number="10">
      <date>2005-02-11</date>
      <customer>AccuData Associates</customer>
    </record>
    - <record id="858" number="9">
      <date>2005-02-12</date>
    </record>
  </SalesRep>
</myReport>

```

## ➤ Zen 리포트를 위한 URI 쿼리 파라미터

브라우저에서 Zen 리포트를 호출할 때 사용할 수 있는 몇 가지 URI 쿼리 파라미터가 제공된다. Firefox 브라우저인 경우에는 모든 파라미터들을 제약없이 자유롭게 사용할 수 있지만, Internet Explorer 경우에는 약간의 제약이 따른다. Zen 리포트 출력에 문제가 발생하면 “Zen 리포트 문제 해결” 장의 “URI 쿼리 파라미터를 사용하여 XHTML 표시하기” 부분을 참고하기 바란다.

다음 테이블에서는 Zen 리포트 URI 쿼리 파라미터들과 대응하는 Zen 리포트 클래스 파라미터를 보여준다.

Zen 리포트를 위한 URI 쿼리 파라미터

URI 쿼리 파라미터	대응하는 클래스 파라미터	설명
\$DATASOURCE	DATASOURCE	Zen 리포트 데이터를 갖고 있는 XML 문서의 URI. 관련 URI는 현재의 URI에 따라 처리된다.
\$EMBEDXSL	EMBEDXSL	1 (true) 또는 0 (false). 이 값이 true이면 Zen은 출력 XHTML에 임베디드된 XSL 명령들을 생성한다. 디폴트 값은 false인데 이 경우 별도의 파일을 생성한다.
\$LOG	—	1 (true) 또는 0 (false). 이 값이 true인 경우, \$MODE=html 또는 \$MODE=pdf를 사용, Zen이 생성하는 중간 파일을 볼 수 있다.
\$MODE	DEFAULTMODE; STYLESHEETDEFAULTMODE 도 참조	\$MODE의 기본 정보는 이미 앞 부분에서 설명하고 있으며 “Zen 리포트 문제 해결”에서 설명하는 것과 같이 \$MODE를 사용, 중간 파일을 볼 수도 있다.
\$NODELETE	—	1 (true) 또는 0 (false). True인 경우 중간 파일을 Caché 임시 디렉터리에 저장한다.
\$REPORTNAME	—	진단 목적을 위해 중간 파일을 저장할 때 사용되는 파일명. \$REPORTNAME는 REPORTNAME은 관계없음.
\$USETEMPFILES	USETEMPFILES	1 (true) 또는 0 (false). True인 경우, 생성된 XSL 파일을 서버의 어플리케이션에 대한 CSP 디렉터리에 저장한다.
\$XSLT	XSLTMODE	파라미터 값 "browser" 또는 "server"는 각각 XSLT를 실행하고, 결과를 생성한다. "browser"가 디폴트 값이다.

\$XSLTVERSION	XSLTVERSION	파라미터 값 “1.0” 또는 “2.0” 은 각각 XSLT로 하여금 리포트를 XSLT 1.0 또는 XSLT 2.0 으로 처리하도록 한다. 디폴트값은 “1.0” 이다.
---------------	-------------	---

## PDF 출력을 위한 Zen 구성

브라우저를 통해 Zen 리포트 클래스를 PDF 형식으로 로딩하는 경우, Caché는 제 3 의 PDF 렌더링 툴(rendering tool)을 작동한다. 렌더링 툴은 XSLT stylesheets를 XML 데이터로 적용하여, XML을 XSL-FO로 전환한 후, XSL-FO를 PDF로 변환한다.

그러나 이 변환 작업은 다음과 같이 필요한 환경 구성하에서만 가능하다:

1. (필수) 다음중 하나의 방법을 사용하여 XSL-FO를 PDF 렌더링 툴(rendering tool)에 설치한다:

- FOP 라는 Apache 공개 소스 프로젝트. 다음 웹 사이트에서 다운로드한 후 설치 문서에 따라 설치한다.  
<http://xmlgraphics.apache.org/fop>
- RenderX 에서 판매하는 XEP 제품. 다음의 RenderX 웹 사이트를 통해 구매할 수 있다:  
<http://www.renderx.com/tools/xep.html>  
 (설치하려면, 키트에 있는 명령(instruction)을 따라 한다.)

2. (필수) 설치된 렌더링 툴(rendering tool)을 호출하는 명령어를 다음 두 가지 중 하나를 사용하여 Zen 리포트 구성에 정의한다.

- 시스템 관리 포탈 [홈] > [환경구성] > [Zen 리포트 설정] 페이지에서 ‘PDF 생성에 이용될 경로 및 파일 이름’에 위에서 설치한 렌더링 툴의 실행 명령을 설정한다. 설정된 구성의 정확성을 검증하기 위해 “지금 검증(Verify)을 수행합니다” 버튼을 통해 설정된 구성을 검증할 수 있다.
- Caché 글로벌 ^%SYS("zenreport","transformerpath")에 직접 렌더링 툴 실행 명령을 정의한다. 예를 들어,

✓ FOP의 경우

```
Set ^%SYS("zenreport","transformerpath")="C:\fop-0.95\fop.bat"
```

✓ XEP의 경우

```
Set ^%SYS("zenreport","transformerpath")="C:\WXEPWXEP.bat"
```

3. (선택) FOP의 경우, 다음의 Apache FOP 웹 사이트의 정보를 토대로 FOP에 대한 사용자 정의 구성 파일을 생성할 수 있다:

[http://xmlgraphics.apache.org/fop \(/0.95/configuration.html\)](http://xmlgraphics.apache.org/fop (/0.95/configuration.html))

사용자 정의 구성 파일을 사용하기 위해 FOP에 대한 Caché callout 이 필요한 경우, 글로벌 ^%SYS("zenreport","transformerconfig") 에 구성 파일 경로에 설정한다. 구성 파일은 FOP에 폰트를 추가하는데 중요하므로, 먼저 폰트 메트릭스를 생성한 후 FOP와 함께 등록한다. 등록 절차는 FOP 웹 사이트를 참조한다.

**참고:** PDF 렌더링은 많은 메모리를 소모하기 때문에, 만약 문제가 발생하면, FOP.bat 또는 XEP.bat 파일을 수정하여, JVM(Java Virtual Machine)의 가용 메모리 양을 늘려 주어야 한다.

## Zen report 문제 해결

여기에서는 다음의 주제들을 대상으로 Zen 리포트 문제 해결을 위한 방법에 대해 설명한다:

- URI 쿼리 파라미터를 사용하여 XHTML 표시하기
- PDF 생성시 발생될 수 있는 문제 해결
- 중간 파일 보기

### URI 쿼리 파라미터를 사용하여 XHTML 표시하기

Zen 리포트 URI 문자열을 입력할 때, \$MODE와 같은 파라미터를 하나만 입력하더라도, 리포트를 생성하기 위해 처리되는 작업에는 내부적으로 추가적인 파라미터들이 존재하기 때문에 다음의 경우와 같이 이러한 추가된 파라미터들을 해독 못하는 문제가 발생하게 된다:

- Zen 리포트 클래스에는 1 을 값으로 하는 CSP 클래스 파라미터 PRIVATE를 갖고 있는데 이 파라미터로 인하여 Zen 리포트 페이지는 전용(private) 페이지가 된다. 전용 페이지는 동일한 CSP 세션에서만 다른 페이지에서 내비게이트된다. 이 경우, Caché는 CSPToken 파라미터를 URI의 쿼리 파라미터 문자열에 자동적으로 추가하게 되는데, Internet Explorer에서 문자열은 추가적인 쿼리 파라미터를 지원하지 않는다.
- Zen 리포트와 관련된 Caché 네임스페이스와 데이터베이스는, [홈] > [보안 관리] > [CSP 응용프로그램] > [CSP 응용프로그램 편집] 페이지에서 “허용된 인증 메소드” 필드의 ‘인증 안함/패스워드’ 옵션을 사용하여, 미인증 액세스(Unauthenticated access) 환경을 구성한다.

미인증 액세스가 설정되면 패스워드 인증이 하지 않는다. 반면 패스워드 액세스는 특정의 Caché 네임스페이스와 데이터베이스와 연관되는 Zen 리포트를 실행하기 전에 필수적으로 사용자 이름과 패스워드를 입력하도록 요구한다. 이러한 로그인 절차를 통하여, Zen은 브라우저의 쿠키 지원을 감지하고, Zen 리포트 URI를 입력하면, 쿼리 문자열에 포함된 복수의 파라미터가 잘 작동되도록 해준다.

로그인 절차가 없는 경우에는, Zen이 브라우저의 쿠키 지원을 감지할 수 없기 때문에, Zen 리포트 URI의 쿼리 문자열에 포함된 추가 파라미터들을 Internet Explorer는 인식하지 못하게 된다.

다음은 이러한 문제들을 해결하기 위해 제시된 방법들이다:

- [홈] > [보안 관리] > [CSP 응용프로그램] > [CSP 응용프로그램 편집] 페이지에서 “세션에 대한 쿠키 사용” 필드 값을 ‘항상’으로 설정한다.
- Zen 리포트 클래스에서 클래스 파라미터 EMBEDXSL=1 을 사용한다.
- Zen 리포트 클래스에서 클래스 파라미터 STYLESHEETDEFAULTMODE="tohtml" 을 사용한다.

## PDF 생성 문제의 해결

PDF 형식으로 Zen 리포트를 출력하는데 발생되는 문제들은 대부분 다음 중 그 원인을 찾을 수 있다.

- XSL-FO를 PDF 렌더링 툴(FOP 또는 RenderX)로 보내는데 필요한 Caché 구성의 부정확성. 구성에 대한 상세한 정보는 “PDF 출력을 위한 Zen 구성”을 참조하기 바란다.
- 렌더링 툴의 설치상 오류.
- 렌더링 엔진이 이해하지 못하는 잘못된 XSL-FO 명령어 사용. Zen은 표준 XSL-FO를 기준으로 XSL-FO를 생성하지만, 모든 렌더링 엔진들이 표준 XSL-FO를 완벽하게 구현하는 것은 아니다. 무료 배포되는 FOP의 경우, 완벽하지 못한 것으로 알려져 있다.
- XData ReportDisplay에서의 문법 오류. 렌더링 툴은 Zen이 서버 상에서 이러한 오류들을 감지하지 못하는 경우 예외로 취급할 수도 있다.

다음의 세션에서는 아래의 환경하에서 Zen 리포트를 PDF 형식으로 출력하는데 수반되는 문제들에 대해 알아본다.

- Zen이 Macintosh 시스템 상의 Caché에서 작동될 때
- 브라우저가 윈도우 시스템에서 실행될 때
- Caché 설치 디렉터리가 /Applications/Cach81 일 때
- 웹 서버가 기본 포트 80으로 구성되었을 때
- Macintosh 서버용 Caché 이름이 mypro 일 때

지금부터의 설명은 FOP가 XSL-FO를 PDF로 변환하는 렌더링 툴이라 가정한다. 그렇지만, RenderX를 사용하는 경우에도 유사한 절차가 적용된다.

1. Caché 터미널을 사용하여, Macintosh 시스템에서의 Caché에 FOP 실행 스크립트를 설정한다:

```
zn "%SYS"
Set ^%SYS("zenreport","transformerpath")="/Applications/fop-0.94/fop"
```

2. 윈도우상의 브라우저에서 리포트를 실행한다:

[http://mypro.local:80/csp/app/ZENApp.MyRep.cls?\\$MODE=pdf](http://mypro.local:80/csp/app/ZENApp.MyRep.cls?$MODE=pdf)

그러면 정상적으로 PDF 리포트는 나타나야 한다.

PDF 리포트가 정상적으로 출력되지 않으면, 다음과 같은 방법으로 FOP 설치상의 문제가 있는지 테스트한다:

1. Caché 서버상의 /Applications/Cach81/mgr/temp 디렉터리가 비어 있는지 확인하다

2. 윈도우즈 브라우저에서 다음 URI를 입력한다.

```
http://mypro.local:80/csp/app/ZENApp.MyRep.cls?$MODE=pdf&$LOG=1&$NODELETE=1
```

3. Caché가 설치된 Macintosh 시스템에서 다음과 같이 XML과 XSL 출력 파일명을 변경한다:

```
cd /Applications/Cache81/mgr/temp
mv *.xsl test.xsl
mv *.xml test.xml
```

4. /Applications/Cach81/mgr/temp 디렉터리에서 다음과 같이 test.xml 과 test.xsl 파일을 사용하여 FOP를 실행한다:

```
/Applications/fop-0.94/fop -xml test.xml -xsl test.xsl -pdf test.pdf
```

5. 콘솔 출력에 오류가 있는지 검사한다. 콘솔 화면에 너무 많은 오류가 나타나면, 다음과 같이 콘솔 출력을 test.log 파일로 다시 지정한다.

```
/Applications/fop-0.94/fop -xml test.xml -xsl test.xsl -pdf test.pdf > test.log
```

6. 출력 파일 test.pdf를 확인한다.

## 중간 파일 보기

특정 URI 쿼리 파라미터를 사용하여 Zen 리포트 문제 해결에 도움을 줄 수 있는데 이러한 파라미터들은 전환 과정에서 생성되는 중간 및 최종 파일을 검토하고 저장할 수 있도록 한다. 이러한 파일들에는 생성된 XSL 또는 XSLFO 파일, XML 파서에서의 오류 메시지를 갖고 있는 텍스트 파일, 최종 XHTML 또는 PDF 파일등이 포함된다.

**참고:** 리포트 옵션을 어떻게 URI 쿼리 파라미터로 제공하는가 하는 방법과 각 브라우저별로 발생할 수 있는 부작용을 처리하는 방법에 대해서는 앞서 “[웹 브라우저에서의 Zen 리포트](#)” 섹션을 참고하기 바란다. 해당 섹션의 “[Zen 리포트를 위한 URI 쿼리 파라미터](#)” 테이블에서는 모든 사용 가능한 URI 쿼리 파라미터를 설명하고 있다.

여기에서는 진단용 쿼리 파라미터들에 대해서만 설명한다: (.)

- \$LOG – \$MODE=html 또는 \$MODE=pdf 를 사용, 진단 메시지를 확인한다.
- \$MODE — 중간 파일중 하나를 검토하기 위한 값(tohtml, toxslfo, xslfo)을 선정한다.
- \$NODELETE — 중간 파일을 Caché 임시 디렉터리에 저장한다.
- \$REPORTNAME — 중간 파일을 사용자 정의 파일명과 위치에 저장한다.
- \$USETEMPFILES — CSP 어플리케이션 디렉터리에 생성된 XSL 또는 XSL-FO 파일을 저장한다.

### ➤ Saxon 메시지 로그에 추가

XHTML 출력을 생성하는 동안 Saxon 파서에 의해 감지되는 오류들을 기록한 텍스트 파일을 위한 진단용 쿼리 파라미터들을 사용하려면, Zen 리포트를 Saxon .jar 파일 위치 정보와 함께 구성하여야 한다. 이러한 구성을 위해 Caché 터미널에서 다음의 명령을 시행한다. Saxon 파서 버전에 따라 설정값에 차이가 있을 수 있다:

```
set ^%SYS("zenreport","saxjar")="c:\Wsaxon65\Wsaxon.jar"
```

또는

```
set ^%SYS("zenreport","saxjar")="c:\Wsaxon\Wsaxon8.jar"
```

또는

```
set ^%SYS("zenreport","saxjar")="c:\Wwaxon9\Wwaxon9.jar"
```

#### ➤ \$LOG 쿼리 파라미터

앞의 “웹 브라우저에서의 Zen 리포트”에서 설명한 바와 같이, \$MODE=html 또는 \$MODE=pdf 를 정의할 때, 쿼리 파라미터 \$LOG=1 를 추가할 수도 있는데, 이 경우 XML의 XHTML로의 변환, 또는 XSLT를 통한 XML의 XSL-FO 과 PDF 로의 각각의 변환 내용을 볼 수 있다. 예를 들면 다음과 같다:

[http://localhost:57772/csp/myPath/myApp.myReport.cls?\\$MODE=pdf&\\$LOG=1](http://localhost:57772/csp/myPath/myApp.myReport.cls?$MODE=pdf&$LOG=1)

여기에서 57772 은 Caché 서버에 지정된 웹 서버 포트 번호이다.

#### ➤ \$MODE 쿼리 파라미터

\$LOG 대신, \$MODE 파라미터에 특정값을 제공함으로써 Zen 리포트에 대한 진단 정보를 표시하는 것도 가능하다. 예를 들면, 다음과 같다:

- XML을 XHTML로 바꾸는 XSLT stylesheet를 보려면,

[http://localhost:57772/csp/myPath/myApp.myReport.cls?\\$MODE=tohtml](http://localhost:57772/csp/myPath/myApp.myReport.cls?$MODE=tohtml)

- XML을 XSL-FO로 바꾸는 XSLT stylesheet를 보려면,

[http://localhost:57772/csp/myPath/myApp.myReport.cls?\\$MODE=toxslfo](http://localhost:57772/csp/myPath/myApp.myReport.cls?$MODE=toxslfo)

- PDF 렌더링 전에 XSL-FO stylesheet를 보려면,

[http://localhost:57772/csp/myPath/myApp.myReport.cls?\\$MODE=xslfo](http://localhost:57772/csp/myPath/myApp.myReport.cls?$MODE=xslfo)

여기에서, 57772 은 Caché 서버에 지정된 웹 서버 포트 번호이다.

#### ➤ \$NODELETE 쿼리 파라미터

\$LOG과 \$MODE은 각각 실시간에 생성된 데이터를 출력할 뿐, 나중에 볼 수 있도록 저장하지는 않는다. 리포트 출력 과정에서, 다양한 유형의 중간 파일들이 생성되는데, \$NODELETE라 불리는 쿼리 파라미터를 추가하여, 모든 중간 및 최종 출력 파일을 저장할 수 있다. Zen은 이러한 파일들의 이름으로 2037q4XM9.xsl 과 같이 자동 생성되는 자동 생성되는 임의의 이름을 부여하게 되는데 필요로 하는 특정 파일은 날짜/시간 및 파일 확장명으로 식별한다.

Zen은 \$NODELETE 파일들을 다음의 위치에 저장하게 된다:

C:\WMyCache\WMgr\WTemp

여기서 C:\WMyCache는 Caché 설치 디렉토리이다.

이 저장 위치는 [총] > [환경구성] > [시작설정] 페이지에서 속성 TempDirectory 값을 수정하여 변경할 수 있는데 Caché는 수정된 디렉토리를 Mgr의 하위 디렉터리로 생성한다.

**참고:** Caché 임시 디렉터리를 변경하는 경우, 이 변경은 Zen 리포트뿐만 아니라 이 임시 디렉토리를 사용하는 모든 Caché 응용프로그램에 적용됨을 유의하여야 한다.

\$NODELETE 파라미터는 \$MODE=html 또는 \$MODE=pdf 일 때 사용할 수도 있고, 출력 저장을 위한 파라미터 \$LOG 또는 \$MODE 와도 함께 사용할 수 있다. 예를 들면 다음과 같다:

```
http://localhost:57772/csp/myns/jsl.MyReport.cls?$MODE=html&$NODELETE=1
```

#### ➤ \$REPORTNAME 쿼리 파라미터

\$REPORTNAME 쿼리 파라미터는 변환 과정에서 생성되는 모든 파일을 저장한다.

**REPORTDIR** 클래스 파라미터에는 Caché 서버 시스템의 로컬 파일 시스템에 이를 파일들을 저장할 위치를 설정한다. REPORTDIR의 값이 주어지지 않을 경우, Zen은 \$REPORTNAME 파일들을 Caché 임시 디렉터리(디폴트, install\_dir\WMgr\Temp)에 저장한다.

앞의 \$NODELETE 쿼리 파라미터에서 설명한 것과 같이, 이 위치는 변경 가능하다. 사용자 출력 파일의 효과적인 관리를 위해, \$REPORTNAME을 사용하려 하는 경우, InterSystems는 REPORTDIR의 값을 설정하도록 권장한다.

**참고:** 대부분의 파라미터들은 \$(달러 표시 문자)를 제외하고는 같은 이름을 공유하지만, REPORTNAME 클래스 파라미터와 \$REPORTNAME 쿼리 파라미터 사이에는 어떠한 관계도 존재하지 않는다.

다음은 \$REPORTNAME 세션의 예이다:

1) Zen 리포트 클래스에 다음과 같이 정의한다:

```
Parameter REPORTDIR = "c:\zenout"
```

2) 브라우저 주소 필드에 아래와 같이 입력한다:

```
http://localhost:57772/csp/rpt/Re.Rpt1.cls?$MODE=html&$REPORTNAME=teste
```

- 57772 은 Caché 서버에 지정된 웹 서버 포트 번호이다.
- Re.Rpt1.cls 는 Zen 리포트 클래스명이다.
- rpt 는 응용프로그램이 정의된 네임스페이스이다.
- teste 는 출력 파일명이다.

3) 1) 에서 설정한 디렉터리로 변경한 후 생성된 파일들을 확인한다:

```
C:\> cd zenout
```

```
C:\zenout> dir
```

```
Volume in drive C has no label.
```

```
Volume Serial Number is 6035-CA91
```

## Directory of C:\zenout

```
06/19/2008 02:55 PM <DIR> .
06/19/2008 02:55 PM <DIR> ..
06/19/2008 02:55 PM 6,320 teste.htm
06/19/2008 02:55 PM 559 teste.txt
06/19/2008 02:55 PM 753 teste.xml
06/19/2008 02:55 PM 4,892 teste.xsl
```

4 File(s) 12,524 bytes

2 Dir(s) 17,536,151,552 bytes free

Saxon .jar 위치를 설정하지 않고도 이와 같은 세션을 실행할 수 있으며 중간 파일들도 확인 가능하다. 그렇지만, 브라우저에서 오류 메시지를 볼 수도 없고, .txt 파일이 생성되지도 않으므로 사용자는 파서로부터 문법 오류에 대한 정보를 얻을 수가 없다.

위와 같은 세션은 \$MODE=pdf 인 경우에도 작동된다. FOP과 RenderX 렌더링 툴은 항상 문법 분석 작업을 하기 때문에, PDF의 경우, 브라우저는 항상 오류 메시지를 출력하며, 사용자는 문법 로그를 갖고 있는 .txt 파일을 항상 볼 수 있다.

### ➤ \$USETEMPFILES 쿼리 파라미터

\$USETEMPFILES 쿼리 파라미터를 1로(true) 설정하면, Zen 리포트 클래스는 생성된 XSLT stylesheet를 파일로 출력한다. 생성된 파일은 출력 XHTML을 생성하는데 사용된다. USETEMPFILES의 디폴트 값은 0 (false)이며, 이 경우 Zen은 XSLT를 생성하고 사용하지만, 파일에 저장하지는 않는다.

\$NODELETE 와 \$REPORTNAME도 많은 옵션을 제공하지만, 생성된 XSLT stylesheet를 출력을 위해 \$USETEMPFILES를 사용하는데에는 다음과 같은 이유들이 있다:

- 스타일 문제를 진단하여야 하는 경우, .xsl 파일이 가장 적절하게 사용된다.
- \$USETEMPFILES는 XSLTMODE 값이 "server"인 경우 발생하는 서버 쪽 비용을 피할 수 있다.
- Zen은 \$USETEMPFILES의 .xsl 파일을 각 Caché 네임스페이스별로 다른 위치에 저장한다. 반면에, \$NODELETE는 모든 중간 파일을 동일한 Caché 임시 디렉터리에 저장한다.
- Internet Explorer 사용자는 브라우저에서 Zen 리포트를 호출할 때, 사용할 수 있는 URI 쿼리 파라미터 수에 제약을 받게 된다. 그래서 브라우저 주소 필드에서 \$NODELETE 또는 \$REPORTNAME를 URI 쿼리 파라미터로 줄 수가 없는 경우가 발생할 수도 있다. 이들과는 달리, \$USETEMPFILES는 USETEMPFILES라는 동등한 Zen 리포트 클래스 파라미터를 가지며 Zen 리포트 클래스에서 이 파라미터값을 1로 설정하여, URI 쿼리 파라미터를 사용하지 않고도

\$USETEMPFILES의 기능을 실행할 수 있다

- 참고: 브라우저에 제공되는 \$USETEMPFILES 쿼리 파라미터는 Zen 리포트 클래스에서 정의된 클래스 파라미터 USETEMPFILES 값을 대체한다.

USETEMPFILES=1 이면, Zen 은 .xsl stylesheet 파일 저장 위치를 다음 글로벌 노드에 설정한다:

```
^%SYS("zenreport","tmpdir")
```

다음의 예은 \$USETEMPFILES 옵션이 설정되어 있을 때, Zen이 생성된 XML을 출력하는 stylesheet 명령이다.

```
<?xml-stylesheet type="text/xsl" href="/csp/myapp/1928b1VL6.xsl" ?>
```

여기에서, *href* 속성의 경로명은 Caché 설치 디렉토리의 **mgr** 하위 디렉토리 **csp**로부터 시작된다. 즉, **csp**는 모든 CSP 응용프로그램의 최상위 디렉토리이며, **myapp**는 어플리케이션이 속한 네임스페이스명(소문자로 변환된)이며 **1928b1VL6.xsl**은 임의로 정의된 .xsl stylesheet 파일명이다.

생성된 파일들을 정기적으로 삭제할 필요가 있는 경우, Caché 터미널이나 ObjectScript 루틴에서 다음 명령어를 실행한다:

```
do ##class(ZENApp.MyReport).%DeleteTempFiles()
```

여기에서, **ZENApp.MyReport**는 Zen 클래스 리포트의 패키지 및 클래스명이다. \$USETEMPFILES 디폴트 값은 0 (false) 이다.

## 제 10 장 Caché Jalapeño

### 1. 자바 Jalapeño Persistence 라이브러리

- 1) 아키텍처
- 2) 설치에 필요한 환경 및 설치

### 2. annotation 사용하기

- 1) Annotation 기능
- 2) 인덱스 추가
- 3) 관계(Relationship) 추가
- 4) 상위 클래스 추가
- 5) 클래스와 프로퍼티 파라미터 추가
- 6) 프로퍼티 타입 변경
- 7) 스키마 생성 제어

### 3. Jalapeño 이클립스(Eclipse) 프러그인

- 1) 프러그인 설정
- 2) Jalapeño 메뉴 참조

## 1. 자바 Jalapeño Persistence 라이브러리

자바 Jalapeño Persistence 라이브러리는 자바 어플리케이션에서 매핑없이 POJO((Plain Old Java Objects)를 저장하고 액세스할 수 있는 가볍고 효율적인 도구이다.

대부분의 데이터베이스 제품들과 프로그램 API 는 SQL 표준 관계형 모델을 기반으로 하고 있기 때문에, 자바 오브젝트를 저장하고 검색하기 위해서는 어떤 형태로 되었던 오브젝트-관계형 매핑을 필요로 하게 된다. 몇몇 표준 오브젝트-관계형 매핑 프레임워크가 있기는 하지만, 프로젝트가 더 복잡하게 되고 관리하기 어려운 단점이 있다.

Jalapeño 는 처음부터 이러한 문제를 예방하기 위해 설계되었다. 사용자는 POJO를 오브젝트-관계형 매핑 없이도 데이터베이스에 저장하고 고성능의 표준 SQL 엔진을 통해 저장된 오브젝트에 쉽게 접근할 수 있다. 또한 Jalapeño 런타임 API 메소드는 보다 용이한 데이터 저장과 검색을 가능하게 한다. 오브젝트 데이터 스키마는 Jalapeño 응용프로그램을 통하여 관계형 데이터베이스에서의 오브젝트를 저장하고 액세스할 수 있게 함으로써, 대응하는 관계형 스키마로 전환 가능하다.

Jalapeño에는 다음과 같은 장점들을 제공한다:

- **용이한 POJO Persistence**

POJO는 persistent 오브젝트로 전환 가능하기 때문에, Jalapeño는 POJO 클래스 정의를 분석하여 대응하는 오브젝트 데이터베이스 스키마를 자동 생성한다. 이렇게 생성된 스키마는 상속, 컬렉션, 관계 및 복합 데이터 형식들과 같은 오브젝트 특성을 모두 사용할 수 있다. 또는 개발자는 자바 소스 파일에 자바 표준 annotation을 추가함으로써, 인덱스, 트리거, 제약등과 같은 추가적인 데이터베이스 기능들을 제공할 수 있다.

- **용이한 오브젝트 액세스**

데이터베이스에서, persistent 오브젝트는 프로퍼티, 관계 및 기타 객체지향 특성을 갖는 진정한 의미의 오브젝트로(오브젝트의 시리얼 상태가 아닌) 저장된다. 응용프로그램에서는, 이러한 POJO 클래스들은 Jalapeño 런타임 API 를 통해 직접 액세스되는데, 이러한 API 들은 데이터베이스 연결, 데이터베이스에서의 오브젝트 검색, 저장, 및 쿼리들과 같은 작업을 수행하는 메소드들을 제공한다.

- **표준 SQL 액세스**

Jalapeño API 는 JDBC API의 상위 집합으로서 구현되기 때문에, 데이터베이스내의 모든 오브젝트는 표준 JDBC API 를 사용하여, 관계형 테이블로서 자동 액세스 가능하다. 따라서 사용자는 새로운 API 의 습득없이도 데이터 쿼리가 가능할 뿐만 아니라 SQL 액세스를 통해 범용 리포트 도구들도 사용할 수 있다.

- 데이터베이스 독립성

Jalapeño는 Caché 와 같은 오브젝트 데이터베이스에서 가장 효과적으로 실행되지만, 한편으로는, 관계형 데이터베이스 기반 어플리케이션을 구현할 수 있도록, SQL 표준 DDL(Data Definition Language)를 사용하여 Jalapeño 오브젝트 데이터베이스 스키마를 관계형 스키마로 전환할 수 있다. 이 때, Jalapeño API 는 데이터베이스 연결을 위해 자동적으로 표준 JDBC 메소드를 사용하며, Caché 오브젝트 데이터베이스에 연결이 되면, 서버 시스템에서는 고성능 오브젝트 기반의 프로토콜이 사용된다.

- 플랫폼 독립성

Jalapeño 라이브러리는 표준 자바를 사용하며, 자바 1.5 이상의 JVM 또는 J2EE 어플리케이션 서버 환경에서 실행된다. Caché 오브젝트 데이터베이스 서버는 거의 모든 OS 플랫폼을 지원하고 있다. Caché 지원 플랫폼에 대한 자세한 정보는 Caché 온라인 문서 “Supported Platforms”을 참조하기 바란다.

## 1) 아키텍처

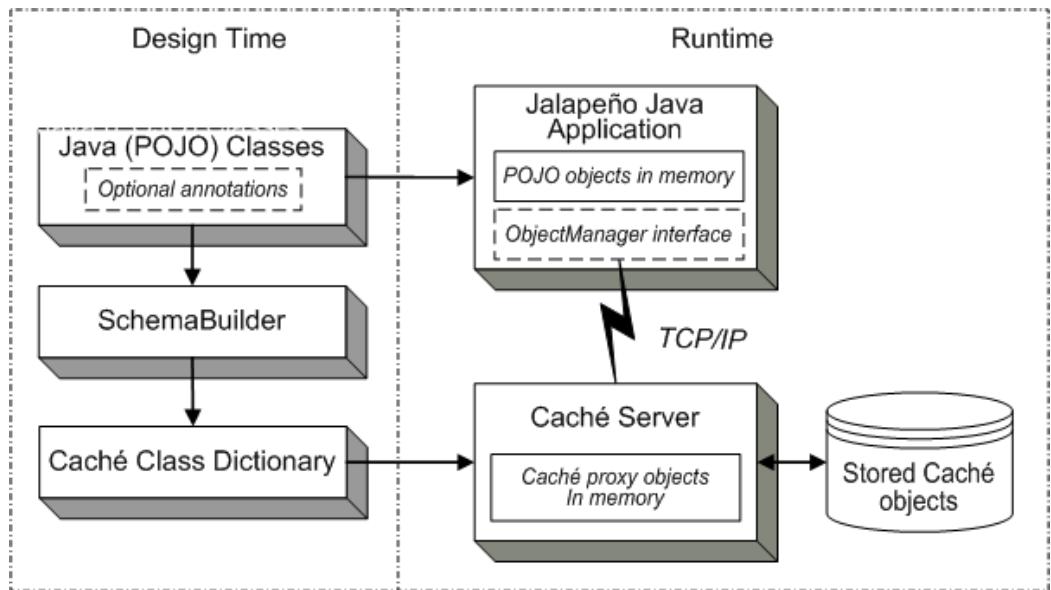
자바 Jalapeño Persistence 라이브러리는 다음과 같은 컴포넌트들로 구성된다:

- **Jalapeño SchemaBuilder** : 자바 클래스를 분석하여 대응하는 오브젝트 데이터베이스 스키마를 생성한다.
- **Jalapeño 런타임 라이브러리**: 데이터베이스 연결, 오브젝트 저장, 검색 및 쿼리를 위한 API 를 제공하는 런타임 자바 라이브러리:
  - ✓ **ObjectManager**: 자바 어플리케이션에서 Caché 데이터베이스 오브젝트에 접근하고 조작하는 메소드를 제공하는 자바 클래스.
  - ✓ **Annotation**: 데이터베이스 구조에 대한 정보를 추가하는 POJO 클래스 코드에서의 옵션 태그 (javadoc 태그와 유사함). 이 정보를 사용하여, SchemaBuilder는 생성된 Caché 클래스에 인덱스나 관계(relationship) 같은 데이터베이스 속성들을 추가한다.

Jalapeño SchemaBuilder는 주어진 자바 클래스들에 대한 Caché 프럭시 클래스 스키마들을 생성하는데 사용된다. 이것은 Caché 클래스로부터 대응하는 자바 프럭시 클래스를 생성하는 표준 Caché 자바 바인딩 아키텍처의 역 기법이라 할 수 있다.

런타임에, POJO 클래스의 인스턴스들은 (TCP/IP 소켓을 사용하여) 대응하는 Caché 서버의 프럭시 클래스의 인스턴스들과 통신한다. 다음은 이 구조를 설명한 그림이다:

## Jalapeño 클라이언트/서버 클래스 아키텍처



## 2) 설치에 필요한 환경 및 설치

자바 Jalapeño Persistence 라이브러리는 Caché 데이터베이스 설치시 함께 설치된다. 그렇지만 자바 1.5 이상의 JVM 환경은 Caché 설치와 별도로 설치되어져야 한다. 이 장에서는 사용자가 다음 항목에 대한 사전 지식이 있음을 전제로 한다:

- 자바 1.5 이상의 JVM 또는 J2EE Application Server 환경
- SQL 또는 관계형 데이터베이스에 대한 경험
- 기초적인 Caché 사용 경험

## 2. annotation 사용하기

오브젝트 데이터베이스 스키마는 자바와 같은 프로그래밍 언어보다 다양한 구조를 표현할 수 있다. 예를 들어, 데이터베이스는 자바 클래스 정의로는 구현할 수 없는 제약, 관계(참조 무결성과 함께), 인덱스와 같은 개념을 정의한다. 뿐만 아니라, 디폴트 오브젝트 데이터베이스 스키마의 특성들을 대체하는데에도 사용할 수가 있다. 예를 들어, Jalapeño SchemaBuilder 가 자동 생성하는 것과는 다른 이름이나 타입을 사용하는 것이 가능하다.

본 장에서는 다음 주제들에 대해 설명하고자 한다:

- **Annotation 기능:** annotation이 데이터베이스 스키마 생성에 미치는 역할 설명.
- **인덱스 추가:** 생성된 Caché 클래스에서 선언되는 인덱스 정의.
- **관계(Relationship) 추가:** 생성된 Caché 클래스에서 선언되는 관계를 명시.
- **상위 클래스 추가:** 생성된 Caché 클래스로 상속되는 상위 클래스 결정.

- **클래스 및 프로퍼티 파라미터 추가:** 생성된 프로퍼티 및 클래스에 사용자 정의 또는 사전 선언된 파라미터 추가.
- **프로퍼티 타입 대체(override):** 프로퍼티 데이터 타입 변경.
- **스키마 생성의 제어:** 클래스와 프로퍼티의 persistent 오브젝트 여부 제어.

## 1) Annotation 기능

Jalapeño SchemaBuilder 유ти리티는 자바 클래스를 분석하여 대응하는 Caché 클래스를 생성한다. 이 때, Caché 클래스를 스키마화 하는데, Caché 데이터베이스 구조에 매핑되는 자바 클래스 데이터의 모델이 된다. 예를 들어, 다음과 같은 자바 클래스에 대해서,

```
package tinyproject;
public class Employee {
    public String name = null;
    public String ssn = null;
    public Department department = null;
}
```

SchemaBuilder는 다음과 같이 위의 자바 클래스에 대응하는 Caché 클래스를 생성한다:

```
Class tinyproject.Employee Extends %Library.Persistent [ SqlTableName = Employee ]
{
    Property name As %Library.String(JAVATYPE = "java.lang.String");
    Property ssn As %Library.String(JAVATYPE = "java.lang.String");
}
```

생성된 정의만으로도 온전한 하나의 Caché 클래스가 되지만, 원래의 자바 클래스에는 포함되어 있 않았던 데이터베이스 구조로 보강한다면 좀 더 유용한 클래스가 될 것이다. 예를 들어, 인덱스를 추가 정의할 필요가 있으면, 자바 클래스 소스에 **com.jalapeno.annotations.Index** 를 **import** 하여 클래스 선언 전에 **@Index annotation** 을 다음과 같이 추가한다:

```
package tinyproject;
import com.jalapeno.annotations.Index;
@Index (name="Empldx", propertyNames={"ssn"}, isPrimaryKey=true)
public class Employee
{...}
```

이렇게 정의된 annotation은 ssn 프로퍼티를 기반으로 하는 “Empldx” 라는 인덱스가 생성되고, 이를 클래스에 대한 기본(Primary) 키로 선언한다. 대응하는 Caché 클래스를 다시 생성하면, 다음과 같이 인덱스 선언이 추가된다:

```
Class tinyproject.Employee Extends %Library.Persistent [ SqlTableName = Employee ]
{
    Property name As %Library.String(JAVATYPE = "java.lang.String");
    Property ssn As %Library.String(JAVATYPE = "java.lang.String");
    Index Empldx On ssn [ PrimaryKey, Unique ];
}
```

위와 같은 용도가 annotation 역할의 전형적인 예라 하겠다. Annotation은 자바 클래스의 원래 정의에 영향을 주지 않고 생성되는 대용 Caché 클래스의 기능을 보강하는데 필요한 지침들을 제공한다.

## 2) 인덱스 추가

위의 예에서, `@Index` annotation을 사용하여, 다음의 인덱스를 정의하였다:

```
@Index (name="Empldx", propertyNames={"ssn"}, isPrimaryKey=true)
```

처음의 두 파라미터는 모든 인덱스 정의에 필요한 필수입력 항목으로서, `name` 은 인덱스명을, `propertyNames` 는 인덱스를 구성하는 프로퍼티들을 정의한다.

`isPrimaryKey` 는 키를 정의하는데 사용되는 다음의 세 파라미터중 하나이다:

- `isPrimaryKey` 는 이 인덱스를 클래스의 SQL 기본 키로 정의한다.
- `isIdKey` 는 이 인덱스를 클래스의 IDKEY로 정의한다.
- `isUnique` 는 이 인덱스를 유일한 인덱스로 정의한다.

### @Indices 사용

`@Indices` 는 클래스에 대한 복수의 인덱스를 정의한다:

```
@Indices({  
    @Index(name="IndexOnName",propertyNames={"name"}),  
    @Index(name="IndexOnSSN",propertyNames={"ssn"})  
})
```

## 3) 관계(Relationship) 추가

데이터베이스 구성에 있어서, 중요한 속성중 하나가 **관계(Relationship)** 이다. 예를 들어, **Person** 과 **Dog** 이라는 두 클래스에 있어서, **Person** 은 여러 **Dog** 을 가질 수 있지만, **Dog** 은 한 명의 **Person** 만을 주인으로 가질 수 있다고 하자. 그러면 다음과 같이 `@OneToMany` 와 `@ManyToOne` annotation 을 사용하여 이들 클래스의 관계를 정의할 수 있다:

```
public class Person {  
    @OneToMany(inverseClass=Dog.class, inverseProperty="getOwner"  
              dependent=true)  
    public ArrayList<Dog> getDogs();  
  
    public class Dog {  
        @ManyToOne(inverseClass=Person.class, inverseProperty="getDogs")  
        public Person getOwner();  
    }  
}
```

`inverseClass` 파라미터는 `Person` 과 `Dog` 이 서로 관계의 멤버라는 것을 명시하며, `inverseProperty` 파라미터는 서로를 참조하는 특수 프로퍼티로서 `Person.getDogs()` 과 `Dog.getOwner()` 를 정의한다. `dependent` 파라미터는 Caché 관계가 parent-child 관계임을 설정한다.

대응하는 Caché 클래스는 다음과 같은 코드를 생성한다:

```
Class tinyproject.Person Extends %Library.Persistent [ SqlTableName = Person ]
{
    Relationship getDog As Dog(JAVATYPE = "java.util.ArrayList")
    [ Cardinality = children, Inverse = getOwner ];
    ...
}

Class tinyproject.Dog Extends (%Library.Persistent) [ SqlTableName = Dog ]
{
    Relationship getOwner As Person(JAVATYPE = "tinyproject.Person")
    [ Cardinality = parent, Inverse = getDog ];
    ...
}
```

**참고:** 역 프로퍼티명은 원래의 소스 자바 클래스가 아니고, 생성된 Caché 프럭시 클래스에서 취한다. 자바와 Caché의 명명 규약의 차이로, 프럭시 클래스에서 생성된 프로퍼티명은 원래의 소스 자바 프로퍼티명과 항상 일치하지 않기 때문이다. 프럭시 클래스 생성시 `@cacheproperty` annotation 을 사용하여 원하는 프로퍼티명을 지정할 수도 있다.

#### 4) 상위 클래스 추가

Caché 클래스 정의중 가장 중요한 속성중 하나는 “`Extends`” 키워드인데, 이는 해당 클래스에 상속되는 상위클래스를 선언하는 키워드이다. 예를 들어서, 모든 persistent Caché 클래스는 `%Library.Persistent` 를 상위 클래스로 선언하여야 한다. 다음은 생성된 Caché 클래스에 상위클래스를 추가하는데 필요한 annotation 에 대한 설명이다:

- `@Extends` : 생성된 Caché 클래스의 기본 클래스로 사용될 클래스를 지정한다.
- `@Embeddable` : 상위클래스 `%Library.SerialObject` 를 추가하고, 파라미터 `[ClassType = serial]` 를 설정함으로써 클래스 시리얼를 만든다.
- `@CacheClass populatable` : 해당 클래스의 샘플 데이터(오브젝트 인스턴스)를 자동 생성하는 메소드를 제공하는 상위클래스 `%Library.Populate` 를 추가한다.
- `@CacheClass xmlSerializable` : 상위클래스 `%XML.Adaptor` 를 추가하여, 클래스를 XML 액세스 가능하게 한다.
- `@Implements` : 생성된 Caché 클래스에 추가할 상위 클래스 목록을 정의한다.

## 기본 상위 클래스 정하기

생성된 Caché 클래스의 상위 클래스는 다음의 규칙에 따라 정해진다:

- 자바 소스 클래스가 `java.lang.Object` 또는 Caché 로 전환되지 않은 자바 클래스를 상속하는 경우에, 디폴트 Caché 상위 클래스는 다음과 같다:
  - Caché ClassType = persistent 이면, `%Library.Persistent`.
  - Caché ClassType = serial 이면, `%Library.SerialObject`.
- 자바 소스 클래스가 Caché로 전환되는 `java.lang.Object` 이외의 다른 자바 클래스를 상속받으면, 전환된 기본 클래스가 디폴트 상위클래스로 사용된다.

## @Extends 사용

`@Extends` annotation 은 생성된 Caché 클래스가 사용할 상위클래스를 명시적으로 지정한다.

다음의 예에서, 자바 클래스 `Test` 는 암묵적으로 `java.lang.Object` 를 확장한다. 생성된 Caché 클래스는 일반적으로 `%Library.Persistent` 를 상위클래스로 갖게 되지만, `@Extends` annotation 은 `%Config.Connectivity` 를 상위 클래스로 갖도록 유도하고 있다:

```
@Extends (Config.Connectivity)
public class Test
{...}
```

생성된 Caché 클래스의 **Extends** 절을 보면, 디폴트 `%Library.Persistent` 선언이 `%Config.Connectivity` 로 대체되어 있는 것을 확인할 수 있다:

```
Class testPkg.Test Extends (%Config.Connectivity)
[ SqlTableName = Employee ]
```

## @Embeddable 사용

`@Embeddable` annotation 을 사용하여 생성된 Caché 클래스를 시리얼 클래스로 지정할 수 있다.

옵션 `access` 파라미터를 사용하여 프로퍼티 액세서(accessors)만이 Caché 클래스의 프로퍼티로 매핑되도록 지정할 수 있다. 다음은 Address 클래스를 serial 형으로 정의하는 예이며,

```
@Embeddable (access = AccessType.PROPERTY)
public class Address{
    public String myField;
    private String mCity;
    public String getCity() {return mCity;}
    public void setCity(String value) { mCity = value;}
}
```

다음은 이에 대응해 생성된 Caché 클래스이다:

```
Class Address Extends %Library.SerialObject
[ ClassType = serial ]
{
  Property City As %Library.String(JAVATYPE = "java.lang.String");
}
```

상위클래스 **%Library.SerialObject** 에는 **Extends** 선언에 포함되었으며, **ClassType** 파라미터값으로 **serial** 이 설정되었다. **City** 프로퍼티만이 생성되었는데, 이는 **myFields** 프로퍼티가 **get** 및 **set** 메소드를 사용하지 않아 무시되었기 때문이다.

#### @CacheClass populatable 사용

**@CacheClass(populatable)** annotation은 **populate()** 와 함께 사용되며 생성되는 Caché 클래스에 상위클래스 **%Library.Populate** 를 추가하여, 테스트 데이터를 자동 생성할 수 있도록 한다.

다음의 코드는 **@CacheClass(populatable)** annotation을 사용하여 **%Populate** 를 상위클래스로서 추가하는 예이다. 클래스 정의에서, 각 프로퍼티는 **POPSPEC** 파라미터를 추가하기 위하여 **@PropertyParameter** annotation을 사용하고 있다:

```
...
@CacheClass(populatable = true)
public class Employee {

  @PropertyParameter(name = "POPSPEC", value = "Name()")
  public String name = null;

  @PropertyParameter(name = "POPSPEC", value = "SSN()")
  public String ssn = null;

  @PropertyParameter(name = "POPSPEC",
  value = "Integer(25000,120000)")
  public int salary = 0;
  ...
}
```

생성된 Caché 클래스에는, **%Library.Populate** 선언이 **Extends** 절에 추가되어 있고, 각 프로퍼티에는 **POPSPEC** 파라미터가 정의되어 있다:

```

Class tinyproject.Employee Extends (%Library.Persistent, %Library.Populate)
[ SqlTableName = Employee ]
{
...
Property name As %Library.String(JAVATYPE = "java.lang.String",
MAXLEN = 4096, POPSPEC = "Name()");
Property salary As %Library.Integer(JAVATYPE = "int",
POPSPEC = "Integer(25000,120000)");
Property ssn As %Library.String(JAVATYPE = "java.lang.String",
MAXLEN = 128, POPSPEC = "SSN()");
...
}

```

#### @CacheClass xmlSerializable 사용

시리얼 오브젝트는 반드시 **%XML.Adaptor**를 상위클래스로 선언하는 클래스에 속해야 한다.

다음은 **@CacheClass(xmlSerializable)** annotation을 사용하여, Employee 클래스를 시리얼화 하는 예이다:

```

@CacheClass(xmlSerializable=true)
public class Employee {
...
}
```

생성된 Caché 클래스에서는, **%XML.Adaptor** 선언이 Extends 절에 포함되어 있는 것을 확인할 수 있다:

```

Class tinyproject.Employee Extends (%Library.Persistent, %XML.Adaptor)
[ SqlTableName = Employee ]
{ ... }
```

#### @Implements 사용

**@Implements** annotation은 디폴트 이외의 추가적인 상위클래스들을 추가하기 위한 상위클래스 목록을 정의한다. 자바의 **implements** 절과는 달리, 추가되는 상위클래스들은 클래스들의 추가적인 기능의 구현을 실제로 포함할 수 있다. 다음은 Employee 클래스에 **%Library.Populate** 와 **%Library.Utility** 를 추가하는 예이다:

```

@Implements({"%Library.Populate", "%Library.Utility"})
public class Employee
{...}
```

Caché 클래스는 이들 추가 상위클래스들을 Extends 절에 포함하고 있다:

```

Class tinyproject.Employee Extends (%Library.Persistent, %Library.Populate,
%Library.Utility) [ SqlTableName = Employee ]
{ ... }
```

## 5) 클래스와 프로퍼티 파라미터 추가

사용자 정의 또는 사전 선언된 파라미터를 생성되는 프로퍼티와 클래스에 추가할 수 있다.

### 클래스 파라미터 추가

`@ClassParameter` annotation은 생성된 Caché 클래스에서 선언되는 클래스 전용 사용자 정의 파라미터를 정의한다. 다음은 생성된 클래스에서 상수로서 사용될 파라미터 `myParam`을 선언하는 예이다:

```
@ClassParameter(name="myParam" value="Hello World")
public class Employee
{...}
```

생성된 Caché 클래스는 다음과 같다:

```
Parameter myParam As STRING = "Hello World";
```

Caché ObjectScript는 정의된 파라미터를 메소드에서 다음과 같이 사용한다:

```
Method writeParam() {
    write ..#myParam
}
```

### `@ClassParameters`

`@ClassParameters`는 다음과 같이 동일 클래스에 복수의 파라미터를 지정할 수 있다:

```
@ClassParameters({
    @ClassParameter(name="myParm1",value="snark"),
    @ClassParameter(name="myParm2",value="boojum")
})
```

### 프로퍼티 파라미터 추가

`@PropertyParameter`을 사용하여 생성된 Caché 클래스의 프로퍼티에 파라미터를 추가할 수 있다:

```
@PropertyParameter (name = "PATTERN", value = "3N1W"-W"2N1W"-W"4N")
public String ssn;

@PropertyParameter (name = "MINVAL", value = "0")
public float balance;

@PropertyParameter (name = "POPSPEC",
value = "ValueList(W",checking,checking,saving,cd,money marketW")")
public String type;
```

### @PropertyParameters

**@PropertyParameters** annotation은 다음과 같이 동일 프로퍼티에 복수의 파라미터를 지정할 수 있다:

```
@PropertyParameters ({  
    @PropertyParameter(name = "PATTERN", value = "3N1W"-W"2N1W"-W"4N"),  
    @PropertyParameter(name = "POPSPEC", value = "SSN()")  
})  
public String ssn;
```

### 6) 프로퍼티 타입 변경

**@CacheProperty(type)** 은 생성된 프로퍼티의 타입을 변경한다. 다음의 예에서, **TreeMap** 프로퍼티는 일반적으로 **%Library.Persistent** 배열로 생성되는데, 여기에서는 **Employee** 오브젝트 배열로 명시적으로 정의하였다:

```
@CacheProperty (type="tinyproject.Employee")  
public TreeMap EmployeeList = new TreeMap();
```

생성된 클래스에서는 프로퍼티 타입은 다음과 같이 정의되었다:

```
Property EmployeeList As array Of tinyproject.Employee (JAVATYPE = "java.util.TreeMap");
```

프로퍼티 파라미터는 종종 재정의된 프로퍼티와 결합된다. 다음은 프로퍼티를 **%Library.Text**로 정의하고 재정의된 타입과 관련된 파라미터들을 설정하는 예이다:

```
@CacheProperty(type="%Text")  
@PropertyParameters ({  
    @PropertyParameter(name = "LANGUAGECLASS", value = "%Text.Japanese")  
    @PropertyParameter(name = "MAXLEN", value = "32000")  
})  
public String notes;
```

### 7) 스키마 생성 제어

Annotation은 대상 클래스와 프로퍼티의 데이터베이스 저장 여부도 제어할 수 있는데, **@Transient** annotation은 클래스나 프로퍼티가 데이터베이스에 데이터를 저장되지 않도록 정의하며, **@Access** annotation은 프로퍼티의 액세스 방법에 따라 프로퍼티의 데이터를 저장할 것인지를 결정한다.

## 클래스와 프로퍼티 저장 않기

@Transient annotation은 클래스 레벨 또는 프로퍼티 레벨로 사용되어, 데이터를 저장하지 않는 클래스나 프로퍼티를 정의한다. 다음은 @Transient를 사용하여, 데이터 저장을 하지 않는 클래스 myClass 를 정의하는 예이며,

```
@Transient()  
public class myClass  
{...}
```

다음은 데이터 저장을 하지 않는 프로퍼티 myProp 를 정의하는 프로퍼티 레벨로서의 @Transient 이다:

```
public class myClass{  
    @Transient()  
    String myProp;  
    public String getMyProp () {  
        getMyProp = myProp;  
    }  
}
```

## 액세스 수준에 따른 지속 프로퍼티 결정하기

@Access(level) annotation을 사용하여 최소한의 액세스 수준을 만족하는 프로퍼티만을 데이터 저장가능 프로퍼티로 정의한다. 다음은 level 파라미터에 대한 유효한 값들의 목록이다:

- AccessLevel.PRIVATE : 모든 프로퍼티의 데이터가 저장된다.
- AccessLevel.PROTECTED : 공용 및 Protect 프로퍼티만 저장한다.
- AccessLevel.PUBLIC : 공용 프로퍼티만 저장한다.
- AccessLevel.UNDEFINED : SchemaBuilder defaultaccesslevel 프로퍼티에 설정되어 있는 디폴트값을 따른다.

디폴트 값은 AccessLevel.UNDEFINED 이다.

다음은 공용 메소드인 PublicFunc 은 지속되지만, 전용 메소드인 PrivateFunc 은 지속되지 않는 예이다:

```
@Access(level = AccessLevel.PROTECTED)  
public class myClass{  
    public void PublicFunc ()  
    {...}  
    private void PrivateFunc ()  
    {...}  
}
```

## 액세스 타입에 따른 지속 프로퍼티 결정하기

@Access(type) annotation은 하나 이상의 액세서(Get 또는 Set 메소드)를 가지는 프로퍼티만을 지속 프로퍼티로 정의한다. 다음은 type 파라미터에 대한 유효한 값들의 목록이다:

- **AccessType.PROPERTY** : getter 와 setter 를 가지는 프로퍼티만 지속시킨다.
- **AccessType.FIELD** : 모든 프로퍼티를 지속시킨다.

SchemaBuilder defaultaccesstype 프로퍼티가 다른 값을 설정하지 않는 한 AccessType.PROPERTY 가 디폴트 값이다.

다음의 예는, 프로퍼티 myProp 가 Get 메소드를 가지고 있기 때문에 지속 프로퍼티로 정의되는 반면에, 캡슐화되어 있지 않은 myField 프로퍼티는 지속되지 않는다:

```
@Access (type = AccessType.PROPERTY)
public class myClass{
    String myField;
    String myProp;
    public String getMyProp () {
        getMyProp = myProp;
    }
}
```

## 3. Jalapeño 이클립스(Eclipse) 프러그인

Jalapeño 이클립스 프러그인은 이클립스에 Jalapeño를 위한 메뉴를 제공한다. 이 메뉴들은 다음과 같은 작업을 수행한다:

- 데이터베이스 서버로의 연결 파라미터 설정 및 연결 제어
- 데이터베이스 스키마 생성 및 설정
- 사용자 코드에 Jalapeño annotation 적용
- 테스트 데이터 생성
- Ant 빌드 파일 및 메타 데이터 생성

### 1) 프러그인 설정

프러그인 설정은 다음 3 단계로 이루어진다:

1. **프러그인 설치**: 프러그인 파일을 이클립스 plugins 디렉터리로 복사한다.
2. **글로벌 라이브러리 정의**: 사용자 JDK와 해당 CacheDB.jar 파일을 글로벌 라이브러리로 정의한다.

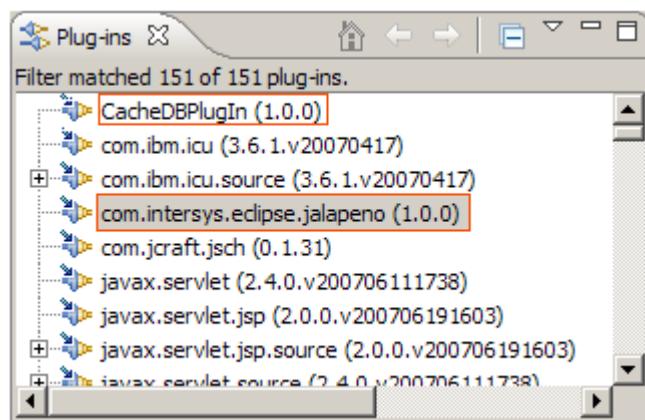
3. Jalapeño 글로벌 프로퍼티 설정: 이클립스의 사용자 프로젝트에서 Jalapeño 옵션 설정을 정의한다.

### 프러그인 설치

Jalapeño 프러그인 설치를 위해서는 이클립스 3.3 이상 버전이 설치되어 있어야 한다. Jalapeño 프러그인 최신 버전은 InterSystems 자바 페이지(<http://www.intersystems.com/java/index.html>)에서 다운로드 받을 수 있다. JDK 1.5 또는 JDK 1.6 사용 여부에 따라 두 가지 프러그인 버전이 있으며, 프러그인은 다음의 두 파일로 구성되어 있다:

- CacheDBPlugIn\_1.0.0.jar
- com.intersys.eclipse.jalapeno\_1.0.0.jar

파일을 복사한 후, 이클립스를 열고, 이클립스 메뉴에서 Window > Show View > Other... 를 선택한다. Show View 창에서 PDE(또는 Plug-in Development) 폴더를 열고 Plug-ins를 선택한다. 설치한 두 개의 파일이 목록에 표시된다(여기서, 목록에 보이지 않을 경우에는 이클립스를 다시 시작한다):



설치가 완료되면 Show View 창에 Jalapeño 폴더가 표시되어야 한다.

### 글로벌 라이브러리 정의

Jalapeño 프로젝트에는 사용자의 자바 JRE 와 참조 라이브러리 CacheDB.jar 파일이 있어야 한다. CacheDB.jar 파일은 다음의 절차에 따라 가져온다:

- 이클립스 메뉴에서 Project > Properties 를 선택한다.
- Properties 창의 왼쪽 목록에서 Java Build Path 를 선택한 후, Libraries 탭을 선택한다
- Add External Jars... 버튼을 클릭하고, <Cache>/Dev/java/lib/<JDK1x> 에 있는 CacheDB.jar를 검색한다. 여기에서, <Cache> 는 Caché 설치 디렉터리이고, <JDK1x> 는 사용하고자 하는 자바 JRE 버전과 일치하는 디렉토리이다. 예를 들어, JRE 1.5를 사용하면 ../JDK15 에 위치한 CacheDB.jar 파일을 선택한다.
- Order and Export 탭을 선택하고, 선택상자를 클릭하여 CacheDB.jar 를 선택한다.

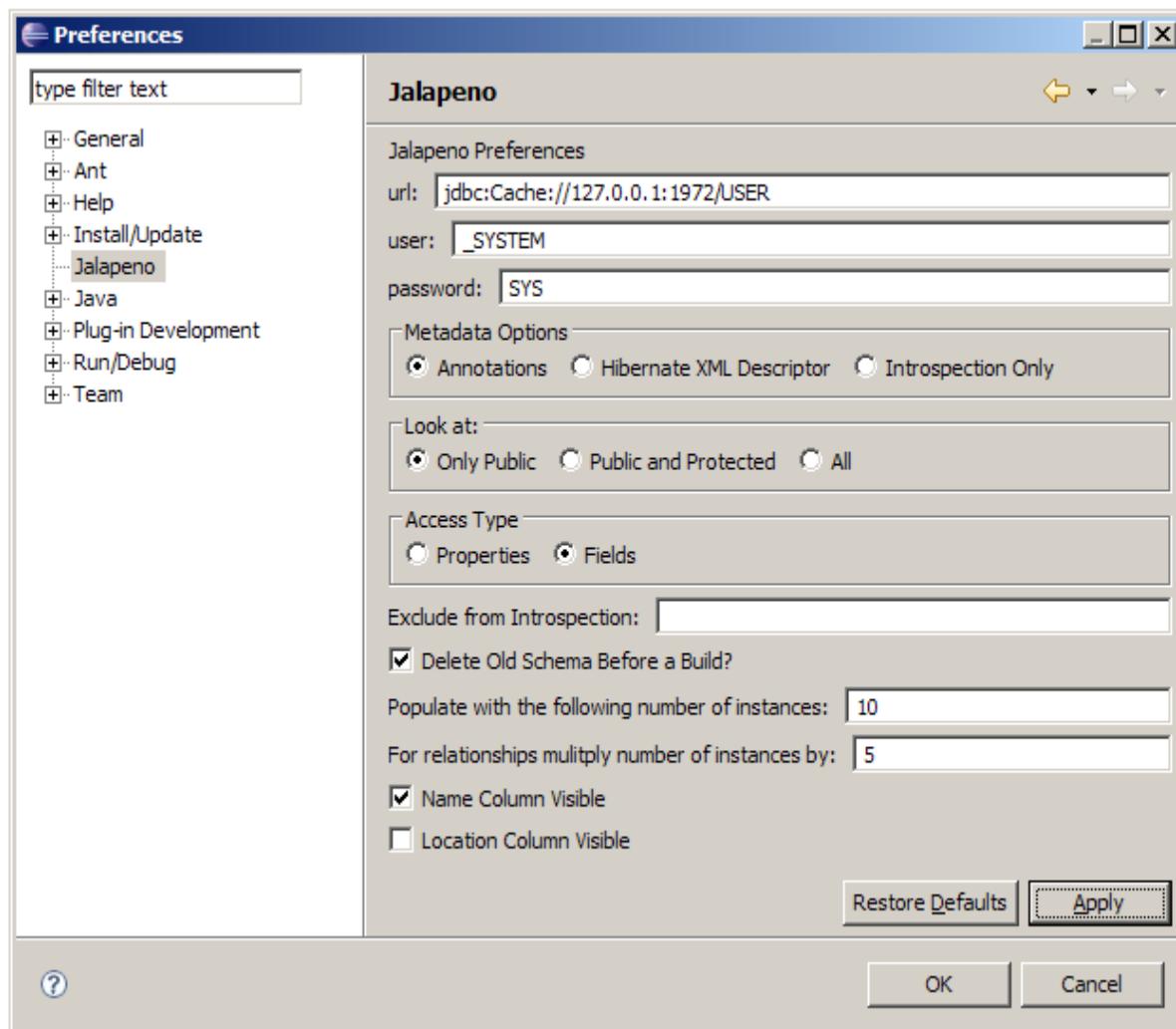
- Properties 창의 Okay (또는 OK) 버튼을 클릭한다.

이제 CacheDB.jar 파일이 프로젝트의 *Referenced Libraries* → *Package Explorer* 뷰 목록에 표시되어져야 한다. (이클립스 3.5 의 경우에는 현재 프로젝트의 Java Resources: src 아래 Libraries 목록에 나타난다).

만약 JRE를 선택하지 않은 경우, *Add External Jars...* 대신 *Add Library...* 를 사용하여, 동일한 절차를 수행한다.

### Jalapeño 글로벌 프로퍼티 설정

Jalapeño 글로벌 설정을 정의하려면, 이클립스 메뉴 **Window > Preferences** 창의 왼쪽 목록에서 Jalapeño를 선택한다. Jalapeño Preferences를 사용하여 연결, 스키마 생성, 테스트 데이터 생성에 대한 글로벌 설정을 할 수 있다:



#### ➤ Jalapeno Preferences

Caché 데이터베이스 연결에 필요한 속성들을 정의하고, 주어진 속성에 기반해 연결 여부를 테스트할 수 있다:

- **url** — 표준 JDBC 연결 문자열
- **user** — 지정된 url 연결 로그인 Caché 사용자명
- **password** — 지정된 url 연결 로그인 Caché 패스워드명

#### ➤ Metadata options

스키마 생성에 필요한 소스를 결정하는 속성들.

- **Annotation** — annotation에 의해 수정된 Introspection 사용.
- **Hibernate XML Descriptor** — Hibernate descriptor 에 의해 수정된 Introspection 사용.
- **Introspection Only** — 수정되지 않은 Introspection 사용.

#### ➤ Look at :

스키마가 introspection을 통해 생성되는 경우, 다음 속성들을 사용하여 지속될 클래스들을 결정한다:

- **Only Public** — Protected 및 전용 클래스는 무시한다
- **Public and Protected** — 전용 클래스는 무시한다.
- **All** — 모든 액세스 레벨의 클래스를 지속 클래스로 정의한다.

#### ➤ Access Type

스키마가 introspection을 통해 생성되는 경우, 다음 속성들을 사용하여 지속될 프로퍼티들을 결정한다:

- **Properties** — get 과 set 메소드를 가지는 프로퍼티만 지속시킨다.
- **Fields** — Public 필드도 지속시킨다.

#### ➤ Exclude from Introspection

생성될 스키마에서 제외되는 패키지의 목록(콤마로 각 패키지 구분).

#### ➤ Delete Old Schema Before Build?

스키마를 새로 생성할 때, 기존의 스키마를 삭제할지 또는 새로운 스키마와 통합할지를 결정한다.

#### ➤ 데이터 관리 옵션

다음의 옵션들은 테스트를 위해 생성할 각 persistent 클래스의 인스턴스 개수를 결정한다:

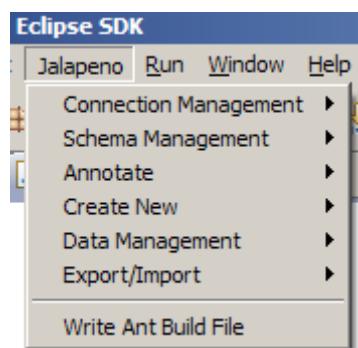
- **Populate with the following number of instances** — 각 persistent 클래스별로 생성할 인스턴스 개수.
- **For relationships multiply number of instances by** — 클래스가 One-to-Many 관계에서 Many 쪽인 경우, 관계의 짹인 상대방의 각 인스턴스에 대해서 생성할 개수.

## 2) Jalapeño 메뉴 참조

Jalapeño 프러그인이 설치되면, 이클립스 메뉴에 **Jalapeño** 메뉴가 추가된다. 이 섹션에서는 이러한 옵션들에 대해 설명한다. 상위 메뉴는 다음과 같다.

### Jalapeno 주 메뉴

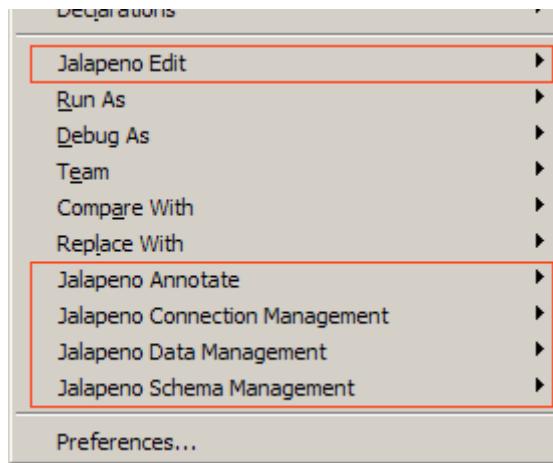
모든 옵션을 포함한다. **Annotate** 하위 메뉴는 Package Explorer 창에서 선택된 항목의 메뉴들이다.



- **Connection Management** — Caché 데이터베이스와의 연결을 관리한다.
- **Schema Management** — Caché 스키마의 생성과 삭제를 관리한다.
- **Annotate** — Package Explorer 창에서 선택한 항목에 대해 Jalapeño annotation을 삽입한다.
- **Create New** — 특수 변수를 생성하고 annotation 을 적용한다.
- **Data Management** — 테스트 데이터의 생성과 삭제를 관리한다.
- **Export/Import** — 클로벌 Jalapeño 설정 내보내기/가져오기.
- **Write Ant Build File** — 사용자 프로젝트용 Ant 빌드 파일을 생성한다.

### Jalapeno 편집 창 콘텍스트 메뉴

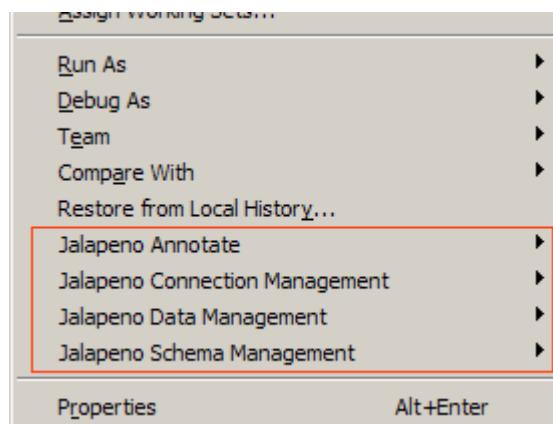
다음은 편집 메뉴로서 annotation 적용에는 항상 **Jalapeno Edit** 를 사용한다.



- **Jalapeno Edit** — 편집 창에서 Jalapeño annotation을 삽입한다.
- **Jalapeno Annotate** — 편집 창에서 Jalapeño 클래스 annotation을 삽입하는 다른 방법.
- **Jalapeno Connection Management** — Caché 데이터베이스와의 연결을 관리한다.
- **Jalapeno Schema Management** — Caché 스키마 생성과 삭제를 관리한다.
- **Jalapeno Data Management** — 테스트 데이터의 생성과 삭제를 관리한다.

#### Jalapeno Package Explorer 콘텍스트 메뉴

다음은 Package Explorer 메뉴로서 Package Explorer 트리에서 선택된 아이템에 annotation을 적용하기 위해 Jalapeno Annotate를 사용한다.

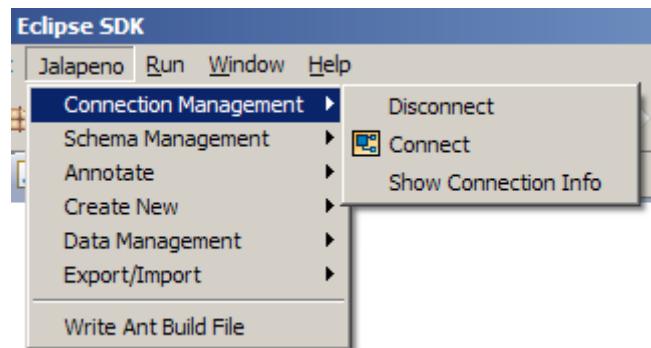


- **Jalapeno Annotate** — Jalapeño annotation을 삽입한다.
- **Jalapeno Connection Management** — Caché 데이터베이스와의 연결을 관리한다.
- **Jalapeno Data Management** — 테스트 데이터의 생성과 삭제 관리한다.
- **Jalapeno Schema Management** — Caché 스키마의 생성과 삭제를 관리한다.

## 연결 관리(Connection Management)

다음은 데이터베이스와의 연결을 관리하는 메뉴로서, 현재 오픈되어 있는 연결 정보를 제공한다.

**Connection Management** 하위 메뉴는 상위 메뉴,Jalapeno 메인 메뉴, Edit Windows 콘텍스트 메뉴, 및 Package Explorer 콘텍스트 메뉴,에서 사용할 수 있다.



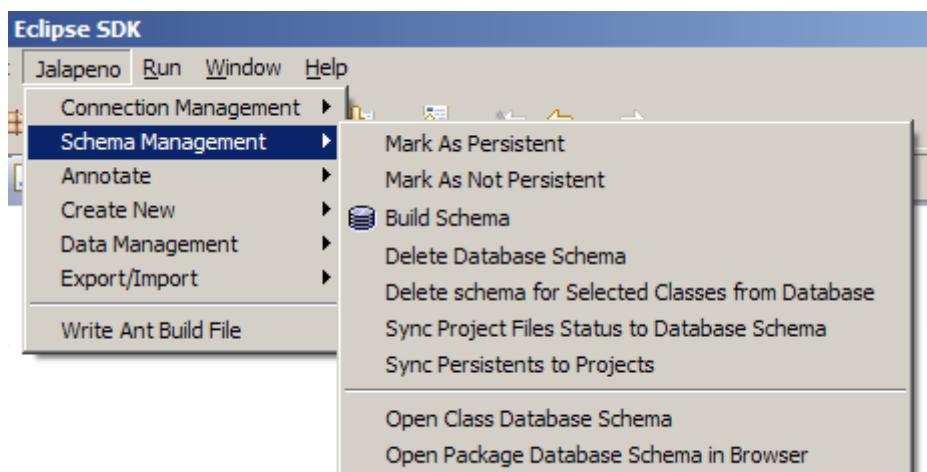
- **Disconnect** – 현재의 연결을 닫는다. 현재 오픈되어 있는 연결이 없는 경우에는 활성화되어 있지 않다.
- **Connect** – Jalapeno Preferences 창에 정의되어 있는 데이터베이스에 연결한다. 현재 오픈되어 있는 연결이 있는 경우에는 활성화되어 있지 않다.
- **Show Connection Info** – 현재 연결되어 있는 데이터베이스의 연결 URI를 표시한다. 예를 들어,

Project TinyPojo is connected to: jdbc:Cache://localhost:1972/USER

현재 오픈되어 있는 연결이 없는 경우, 이 항목은 활성화되어 있지 않다.

## 스키마 관리(Schema Management)

데이터베이스 스키마 생성을 제어하는 옵션으로서, 스키마 관리 하위메뉴는 상위 메뉴,Jalapeno 메인 메뉴, Edit Window 콘텍스트 메뉴, 및 Package Explorer 콘텍스트 메뉴,에서 사용할 수 있다.



- **Mark As Persistent** – 이 옵션으로 표시되는 클래스는 프로젝트에 대한 데이터베이스 스키마가 생성될 때마다 항상 포함된다. 표시된 클래스는 Project Tool Window 의 Cache Cube() 에 나타난다. 예를 들어, **Employee** 와 **Department** 클래스는 TinyPojo 샘플 프로그램에서 persistent로 표시되어 있다.
- **Mark As Not Persistent** – 데이터베이스 스키마에 포함된 클래스 목록에서 표시된 클래스들을 제거한다.
- **Build Schema** – persistent로 표시되는 클래스에 대해 새로운 Caché 데이터베이스 스키마를 생성한다. 만약 Jalapeno Preferences 창에서 "Delete Old Schema Before Build?" 옵션이 체크되어 있으면, 기존의 스키마는 새로운 스키마 생성 전에 삭제된다.
- **Delete Database Schema** – 현재의 Caché 데이터베이스 스키마의 모든 클래스들을 삭제한다.
- **Delete Schema for <Selected Classes> from Database** – 현재의 데이터베이스 스키마에서 선택된 클래스들을 삭제한다.
- **Sync Project Files Status to Database Schema** – 데이터베이스 스키마에서 persistent Caché 클래스로서 표시된 프로젝트 파일들의 동기화를 시도한다.
- **Sync Persistents to Project** – 현재 오픈 프로젝트의 Show Persistents 창에 표시되는 클래스들을 동기화한다. (즉, 오픈 프로젝트에 없는 클래스들과, 디스크에서는 삭제되었지만 프로젝트에서 간신히 찾을 수 있는 클래스들은 목록에서 없앤다)
- **Open Class Database Schema** – 시스템 관리 포탈에서 해당 클래스에 대응하는 SQL 테이블을 표시한다.
- **Open Package Database Schema in Browser** – 시스템 관리 포탈에서 해당 패키지의 클래스 목록을 포함하는 SQL 테이블을 표시한다.

## Annotation

Annotation 옵션은 메인 메뉴의 **Annotation**이나, 콘텍스트 메뉴의 **Jalapeno Edit** 및 **Jalapeño Annotate**에서 액세스할 수 있다. 편집 창에서 선정된 항목에 대해 annotation을 적용할 때에는, 항상 **Jalapeno Edit**를 사용한다. 다음의 두 하위 메뉴는 **Package Explorer** 창에서 선택되는 항목에 대해서만 실행된다.

```
Class As Populatable  
Class As Transient  
Class As Embeddable  
Create ID Property  
Create Version Property  
Add Full CacheClass Annotation  
Access Type Properties (getters and setters)  
Access Type Fields  
Access Level Public  
Access Level Protected  
Access Level Private  
Property As Id  
Property As Primary Key  
Property As Transient  
Property As Unique  
Property As Version  
Property As Establish Relationship  
Property With Constraint Min/Max  
Property With Constraint Pattern  
Property With POPSPEC  
Property Add Full Cache' Property Annotation
```

```
Class As Populatable  
Class As Transient  
Class As Embeddable  
Create ID Property  
Create Version Property  
Add Full CacheClass Annotation  
Access Type Properties (getters and setters)  
Access Type Fields  
Access Level Public  
Access Level Protected  
Access Level Private  
Property As Id  
Property As Primary Key  
Property As Transient  
Property As Unique  
Property As Version  
Property As Establish Relationship  
Property With Constraint Min/Max  
Property With Constraint Pattern  
Property With POPSPEC  
Property Add Full Cache' Property Annotation
```

#### 클래스 annotation:

- **Class As Populatable** — `populatable` 파라미터를 true 로 설정하여 `@CacheClass` 를 삽입한다. 예를 들어, 다음과 같이 클래스 `Employee` 의 정의에 포함된다:

```
@CacheClass(populatable = true)  
Public class Employee {
```

- **Class As Transient** — `@Transient` 삽입:

```
@Transient  
public class Employee {
```

- **Class As Embeddable** — `@Embeddable` 삽입:

```
@Embeddable  
public class Employee {
```

- **Create ID Property** — ID 표시자 프로퍼티를 추가하고, `@ID`로 annotation 적용.
- **Create Version Property** — Version 프로퍼티를 추가하고, `@Version` 으로 annotation 적용.
- **Add Full @CacheClass Annotation** — 모든 파라미터를 명시적으로 디폴트값으로 설정하여, `@CacheClass` 를 삽입한다:

```
@CacheClass(name = "", packageName = "", sqlTableName = "", sqlSchemaName = "",  
javaProjectionClassName = "", javaProjectionPackageName = "", populatable = false,  
xmlSerializable = false)  
public class Employee {
```

- Access Type <Properties/Fields> — type 파라미터를 Properties 또는 Fields로 설정하여, @Access를 삽입한다:

```
@Access(type = AccessType.PROPERTY)
public class Employee {
```

옵션값:

Properties (getters 및 setters) — 파라미터 type은 AccessType.PROPERTY로 설정된다.

Fields — 파라미터 type은 AccessType.FIELD로 설정된다.

- Access Level <Public/Protected/Private> — level파라미터를 Public, Protected, 또는 Private으로 설정하여 @Access를 삽입한다:

```
@Access(level = AccessLevel.PUBLIC)
public class Employee {
```

옵션값:

Public — 파라미터 level은 AccessLevel.PUBLIC으로 설정된다.

Protected — 파라미터 level은 AccessLevel.PROTECTED로 설정된다.

Private — 파라미터 level은 AccessLevel.PRIVATE으로 설정된다.

프로퍼티 annotation:

- Property As Id — @ID를 삽입한다:

```
@ID(type=IDType.SYSTEM_ASSIGNED)
public String myprop;
```

- Property As Primary Key — isPrimaryKey파라미터를 true로 설정하여, @Index를 삽입한다. 하나의 @Index인스턴스만을 삽입하는 경우에도 @Indicesannotation으로 @Indexannotation을 감싸게 된다. @Index는 클래스-레벨annotation이기 때문에, 텍스트는 클래스 선언 이전에 첨부되어져야 한다:

```
@Indices({@Index(name="MypropIdx", propertyNames={"myprop"}, isPrimaryKey=true)})
public class Employee {
...
public String myprop;
```

- Property As Transient — @Transient를 삽입한다:

```
@Transient
public String myprop;
```

- Property As Unique — isUnique파라미터를 true로 설정하여, @Index를 삽입한다. 하나의 @Index인스턴스를 삽입하는 경우에도 @Indicesannotation으로 @Indexannotation을

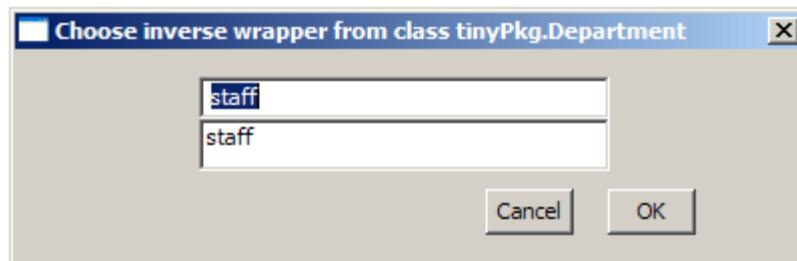
감싸게 된다. @Index 는 클래스-레벨 annotation 이기 때문에, 텍스트는 클래스 선언 이전에 첨부되어져야 한다:

```
@Indices({@Index(name="MypropIdx", propertyNames={"myprop"}, isUnique=true)})  
public class Employee {  
    ...  
    public String myprop;
```

- **Property As Version** — @Version 을 삽입한다:

```
@Version  
public String myprop;
```

- **Property As Establish Relationship** — @Relationship 을 삽입한다. 이 옵션을 선택하게 되면, 선택 가능한 역(inverse) 프로퍼티들의 목록이 나타나므로, 대응하는 inverseProperty 파라미터를 선택할 수 있다:



```
@Relationship(type=RelationshipType.MANY_TO_ONE,  
    inverseClass="tinyproject.Department", inverseProperty="staff")  
public Department department = null;
```

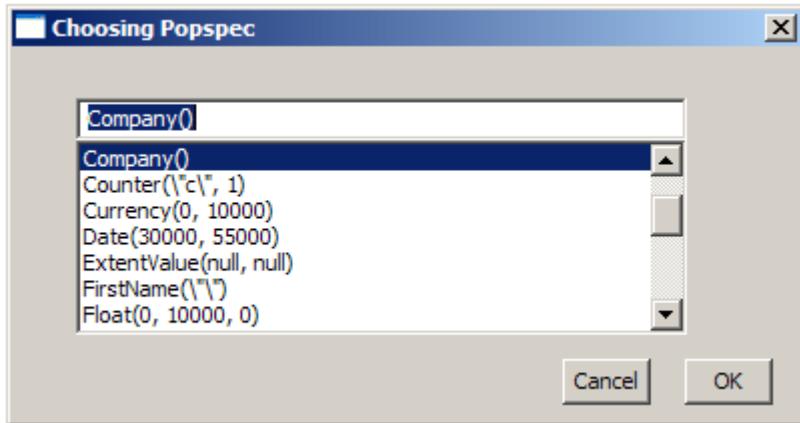
- **Property With Constraint Max/Min** — name 파라미터를 MINVAL 과 MAXVAL 로 설정하여, 두 개의 @PropertyParameter 인스턴스를 삽입한다:

```
@PropertyParameters({@PropertyParameter(name="MINVAL", value=""),  
    @PropertyParameter(name="MAXVAL", value="")})  
public String myprop;
```

- **Property With Constraint Pattern** — name 파라미터를 PATTERN 으로 설정하여, @PropertyParameter 를 삽입한다. 하나의 @PropertyParameter 를 삽입하는 경우에도, @PropertyParameters annotation 으로 @PropertyParameter annotation 을 감싸게 된다:

```
@PropertyParameters({@PropertyParameter(name="PATTERN", value="")})  
public String myprop;
```

- **Property With POPSPEC** — name 파라미터를 POPSPEC 으로 설정하여, @PropertyParameter 를 삽입한다. 이 옵션을 선택하는 경우, Choosing Popspec 드롭다운 리스트에서 annotation 에 대한 파라미터 값을 선택할 수 있다:



```
@PropertyParameters({@PropertyParameter(name="POPSPEC", value="Company()")})
public String myprop;
```

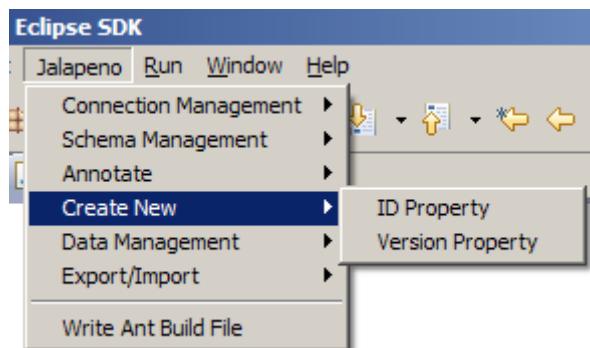
이 annotation 을 갖는 클래스는 클래스 annotation @CacheClass(populate = true) 도 함께 포함하여야 한다. (**Class As Populatable** 메뉴 옵션 참조.)

- **Property Add Full Cache' Property Annotation** — 모든 파라미터를 명시적으로 디폴트값으로 설정하여, **@CacheProperty** 를 삽입한다:

```
@CacheProperty(name = "", type = "", required = false, sqlColumnName = "")
public String myprop;
```

### 새로 생성하기( Create New )

이 옵션을 통해, 클래스에 annotation을 적용하는 프로퍼티를 추가한다. Jalapeno 메뉴의 하위 메뉴인 **Create New** 또는 콘텍스트 메뉴의 하위메뉴 **Jalapeño Annotate** 와 **Jalapeno Edit** 를 통해 옵션을 선택한다.



#### ➤ ID 프로퍼티 생성(ID Property)

현재 클래스의 첫 머리에, 내부 데이터베이스 ID annotation을 삽입한다. 다음은 **@ID** annotation 과 **IdPlaceholder** 프로퍼티를 Employee 클래스에 추가하는 예이다:

```
public class Employee {  
    @ID(type = IDType.SYSTEM_ASSIGNED)  
    private String idPlaceHolder;
```

#### ➤ 버전 프로퍼티 생성(Version Property)

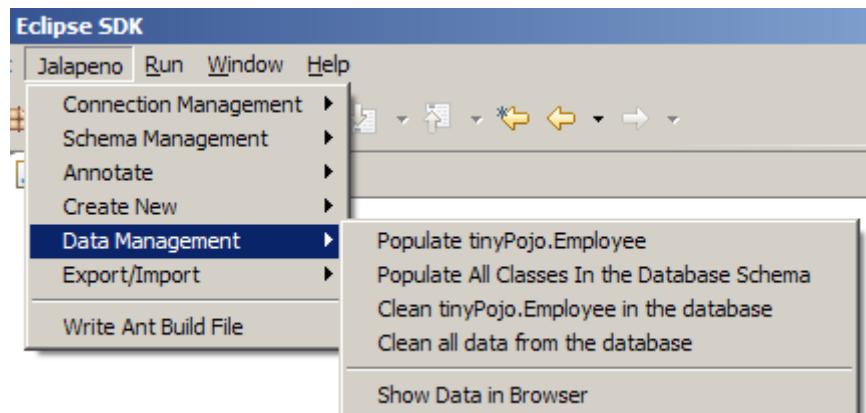
현재 클래스의 첫 머리에, 버전 정보를 갖는데 사용되는 Version annotation을 삽입한다.

다음은 **@Version** annotation 과 **long\_Version** 프로퍼티를 Employee 클래스에 추가하는 예이다:

```
public class Employee {  
    @Version  
    private long _Version;
```

## 데이터 관리(Data Management)

이 옵션은 데이터베이스에서의 테스트 데이터 생성을 제어한다. 데이터 관리 옵션은 세 개의 상위 메뉴, Jalapeno 메인 메뉴, Edit Window 콘텍스트 메뉴, Package Explorer 콘텍스트 메뉴,에서 사용할 수 있다.

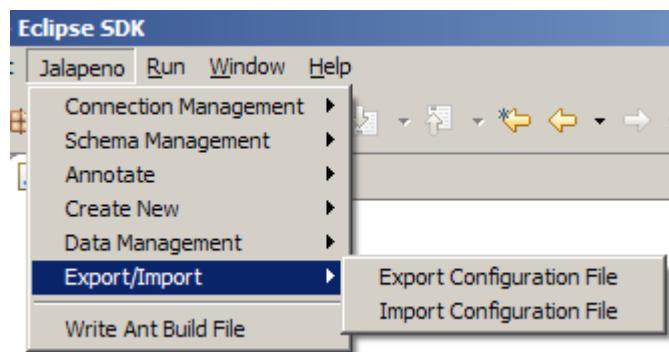


- **Populate <selected classes>** - Project Tool Window에서 선택한 클래스에 대한 테스트 데이터를 생성한다.
- **Populate All Classes In the Database Schema** - 현재의 데이터베이스 스키마에 속한 모든 클래스들에 대한 테스트 데이터를 생성한다.
- **Clean <selected classes> in the database** - 선택한 클래스에 대한 모든 데이터를 데이터베이스에서 삭제한다.
- **Clean all data from the database** - 모든 데이터를 현재의 데이터베이스에서 삭제한다.
- **Show Data in Browser** - 시스템 관리 포탈에서 해당 클래스에 대응하는 SQL 테이블을 보여준다.

## 내보내기/가져오기(Export/Import)

다른 IDE에서 생성된 XML 파일로부터 Jalapeño 프로젝트 정보를 가져온다. 해당 파일은 Jalapeno Preference 창에서 정의된 프로퍼티들로서 Jalapeño 메타 데이터를 갖고 있다. Export/Import 하위 메뉴는 Jalapeno 주 메뉴에서만 사용 가능하다.

**Export Configuration File** 및 **Import Configuration File** 옵션은 항상 활성화되어 있는데, 이것은 Jalapeño 환경 설정 항목들이 Jalapeno Preferences 창에서 포괄적으로 정의되고 있기 때문이다. 해당 preferences들은 프로젝트 단위로 정의되는 것이 아니고 작업 공간(workspace) 전체에 영향을 미치고 있다. 개별적으로 preference를 구성하려면, 해당하는 Jalapeño annotation을 사용한다.



### • 환경 설정 파일 내보내기 (Export Configuration File)

다음은 Jalapeño 이클립스 플러그인을 통해 TinyPojo 프로젝트에서 생성된 샘플 프로젝트 파일이다:

```
<?xml version="1.0" encoding="UTF-8"?>
<jalapeno-configuration version="1">

<project name="TinyPojo">
    <persistent-classes>
        <class>TinyPojo.Department</class>
        <class>TinyPojo.Employee</class>
    </persistent-classes>
</project>

<import-options
    access-type="fields"
    access-level="public"
    use-hibernate-descriptors="false">
    <exclusions></exclusions>
</import-options>

<build-options merge-on-build="false">
</build-options>
```

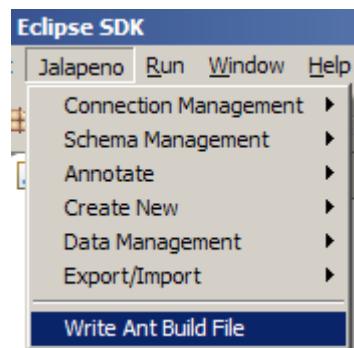
```
<refactoring-options pull-index-annotation="false">  
</refactoring-options>  
  
<ide-warnings enabled="false">  
</ide-warnings>  
  
</jalapeno-configuration>
```

- **환경 설정 파일 가져오기 (Import Configuration File)**

다른 Jalapeño IDE 프러그인에 의해 생성된 구성 파일을 가져온다.

#### Ant 빌드 파일 작성(Write Ant Build File)

프로젝트에 대한 표준 Ant 빌드 파일을 생성한다. 이 옵션은 Jalapeno 주 메뉴에서만 사용할 수 있다.



파일은 표준 빌드 정보와 (Jalapeno Preferences 창에서 정의한) Jalapeño 메타 데이터를 모두 포함하고 있다.

# Caché 오류 참조

## 1. 시스템 오류 메시지

- 1.1 일반적인 시스템 오류 메시지
- 1.2 색도잉 오류 메시지
- 1.3 ANSI 표준 M 오류 메시지

## 2. SQL 오류 메시지

## 3. 개체 오류 메시지

## 4. CSP 오류

## 시스템 오류 메시지

다음 표에서는 Caché 시스템 오류 메시지 목록입니다. 시스템 프로세스가 오류로 종료하면 운영자 콘솔 기능에서 오류 메시지를 게시합니다.

시스템 오류 메시지 - A에서 E

오류 코드	설명
<ALARM>	사용자 이벤트에 대한 내부 타이머가 만료되었습니다.
<ARRAY DIMENSION>	변수 또는 인수 예상되는 차원이 잘못되었습니다.
<BAD IMPLICIT>	잘못된 암시적 데이터 변환이 요구되고 있습니다.
<BLOCKNUMBER>	데이터베이스 파일의 범위를 블록을 참조했습니다
<_CALLBACK SYNTAX>	(오류 코드 이름 밑줄주의하십시오.) 밑줄 문자로 시작 문자가 계속 이름이 지정되어 있습니다.
<CANNOT GET THIS PROPERTY>	클래스의 속성을 가져 오려고했지만, 이 속성의 취득은 잘못
<CANNOT SET THIS PROPERTY>	클래스의 속성을 설정하려고했지만, 이 속성의 설정은 잘못
<CLASS COMPILING>	로컬 시스템에서 현재 재컴파일되는 클래스의 인스턴스를 생성하거나 클래스의 클래스 메소드를 호출하려고했습니다
<CLASS DESCRIPTOR>	클래스 기술자이다 루틴을 실행하려고했습니다
<CLASS DOES NOT EXIST>	존재하지 않는 클래스를 참조했습니다
<CLASS EDITED>	객체를 생성한 후에 클래스가 원격 시스템에서 컴파일되는 로컬 시스템에 있는 객체를 사용하려고했습니다
<CLASS RECOMPILED>	객체를 생성한 후에 클래스가 로컬로 컴파일되는 로컬 시스템에 있는 객체를 사용하려고했습니다
<CLASS TOO BIG TO LOAD>	클래스 기술자가 너무 커서 일과 버퍼에 맞지 않기 때문에 클래스를 사용할 수 없습니다
<CLASS TOO BIG TO	클래스 기술자가 너무 커서 일과 버퍼에 맞지 않기 때문에 클래스를

SAVE>	생성할 수 없습니다.
<CLIENT-SERVER MISMATCH>	네트워크 요청은 클라이언트와 서버 사이에 호환성이 없기 때문에 처리할 수 없습니다
<CLUSTERFAIL>	글로벌 버퍼 잠금 과정에서 클러스터 구성원이 실패했습니다
<COLLATECHANGE>	첨자있는 지역 변수가 정의되어있을 때, 일치 알고리즘을 변경하려고했습니다.
<COLLATEMISMATCH>	잘못된 데이터 유형 구성을위한 하위 스크립트 레벨 매핑에 실패했습니다
<COLLATION NOT SUPPORTED>	현재 시스템에서 지원되지 않는 데이터 타입의 글로벌을 참조했습니다
<COMMAND>	루틴에 인자가없는 <b>GoTo</b> 같은 상황에 잘못된 명령을 사용했습니다
<COMMITFAIL>	<b>TCommit</b> 과정에서 Caché 받은 오류를 COMMIT 사이 받습니다. 이 오류는 여러 원격 컴퓨터가 실제로 커밋을 처리했는지 Caché 가 인식하지 않는다는 것을 의미합니다.
<COMPLEX PATTERN>	패턴을 입력 문자열의 결합에 의한 수있는 일치가 너무 많아서 관리할 수 없습니다.
<CONFLICTING BLOCK NUMBERS>	이미 예약된 블록을 예약하려고했습니다
<CORRUPT OBJECT>	내부 개체의 시스템 오류가 발생했습니다. 이 오류가 발생했을 경우, 인터 시스템즈의 기술 지원 서비스에 문의하십시오.
<CORRUPT VOLUME SET>	볼륨 세트가 손상되었습니다. 일반적으로 볼륨 세트의 레이블이 부정한 것을 의미합니다. 수정하려면 LABEL 유틸리티를 사용하십시오.
<CP NOT STARTED>	시스템의 적절한 처리에 필요한 주요 프로세스 1 개를 시작하지 못했습니다. 이것은 아주 심각한 시스템 오류가 발생할 수 있기 때문에 시스템 관리자에게 알리십시오.
<DATABASE MAP LABEL>	데이터베이스 맵 블록에 잘못된 레이블이 있습니다
<DATABASE>	Caché 데이터베이스의 손상을 발견했습니다 (이것은 아주 심각한

	시스템 오류가 발생할 수 있기 때문에 시스템 관리자에게 알리십시오)
<DDP JOB OVERFLOW>	Caché 작업의 내부 작업 번호가 1544 을 초과하고 있지만, DSM 데이터베이스에 DDP 를 사용하여 액세스하려고합니다. 이 작업 번호는 DDP 처리하려면 너무 큽니다.
<DIRECTORY>	대상 시스템과 Caché 데이터베이스에 해당 디렉토리가 없습니다. Caché 데이터베이스가 탑재되지 않았거나 데이터베이스가 다른 구성으로 잠겨 있습니다.
<DISCONNECT>	장기 요청 중에 TCP 의 연결이 감지되었습니다
<DISKHARD>	Caché 는 수정 불가능한 디스크 하드웨어 오류가 발생했습니다 (이 경우 데이터베이스 오류가 발생할 수 있기 때문에 시스템 관리자에게 알리십시오)
<DIVIDE>	0 으로 나누기하려고했습니다
<DOMAINSPACERTRY>	도메인 공간 마스터 반복적으로 연결하려했지만 실패했습니다
<DSCON>	연결되지 않는 터미널에서 읽으려고했습니다
<DSKFUL>	파일이 최대 크기에 도달했기 때문에, 디스크 파일에 데이터를 쓰는데 실패했습니다. 일부 데이터를 기록했지만, 모든 데이터가 기록되지 않습니다.
<DUPLICATEARG>	이미 정의된 \$ SORTBEGIN 전역의 조상이나 후손과 함께 \$ SORTBEGIN 을 사용하려고했습니다
<DYNAMIC LIBRARY LOAD>	설명선 통해 동적 라이브러리를 로드하는 중 오류가 발생했습니다. 자세한 내용은 " <a href="#">cconsole.log</a> "를 참조하십시오.
<ECODETRAP>	\$ ECODE 시스템 변수에 NULL 이 아닌 문자열 값을 설정함으로써 사용자 생성 소프트웨어 함정이 생성되었습니다
<ENDOFFILE>	시퀀스 파일의 EOF 마커를 넘어 읽으려고했습니다
<ERRTRAP>	나머지 시스템 리소스가 부족하기 때문에 오류 트랩 시저를 실행할 수 없습니다.
<EXTERNAL INTERRUPT>	다른 프로세스가 이 과정에 간섭을 끼쳤습니다.

## 시스템 오류 메시지 - F에서 J

오류 코드	설명
<FILEFULL>	Caché 을 글로벌 데이터와 루틴 스토리지의 디스크 블록을 할당하려고했습니다. 그러나 Caché 데이터베이스는 꽉 확장할 수 없기 때문에 그 할당은 실패했습니다.
<FRAMESTACK>	<b>Do,For,Xecute,New</b> 사용자 정의 함수 등 루틴에 중첩 호출이 너무 많습니다
<FUNCTION>	지정된 함수가 존재하지 않거나 사용이 잘못되었습니다
<GARBAGE COLLECTOR FAILED>	데이터베이스 공간을 회수하는 과정 1 개가 실패했습니다. 이것은 아주 심각한 시스템 오류가 발생할 수 있기 때문에 시스템 관리자에게 알리십시오.
<HALTED>	내부적인 오류 메시지입니다
<ILLEGAL VALUE>	$\$ X$ 와 $\$ Y$ 와 같이 허용되지 않는 장소에서 음수를 사용하려고했습니다
<INSUFFICIENT CLASS MEMORY>	Caché 공유 메모리가 부족하기 때문에, 클래스를 사용할 수 없습니다
<INTERNAL OBJECT ERROR>	내부 개체의 시스템 오류입니다. 이 오류가 발생했을 경우, 인터시스템즈의 기술 지원 서비스에 문의하십시오.
<INTERRUPT>	사용자가 일과를 중단했습니다 (대부분의 경우 사용자가 <b>CTRL - C</b> 키를 눌러 있습니다.)
<INVALID ARGUMENT>	설명선 함수 <b>zfentry</b> 지정 잘못된 인수 프로토 타입이 있습니다
<INVALID BIT STRING>	비트 문자열 처리에 사용되는 비트 문자열이 올바르지 않습니다
<INVALID CLASS>	깨진 클래스를 사용하려고했습니다. 클래스를 다시 컴파일하고 다시 시도하십시오.
<INVALID FILE VARIABLE>	파일 변수가 필요했지만, 지정되지 않았습니다.

<INVALID OREF>	현재 메모리에 지정된 OREF 을 가진 개체가 존재하지 않습니다
<INVALID SELECT LIST>	SELECT 목록이 필요했지만, 지정되지 않았습니다.
<INVALID TYPE>	OREF 이 허용되지 않는 곳에 사용되고 있습니다
<Java Exception>	Java 런타임 환경에서 호출하는 동안 예외가 발생했습니다
<Java VM not loaded>	사용 가능한 Java 가상 머신이 없습니다

#### 시스템 오류 메시지 - K에서 O

오류 코드	설명
<LABELREDEF>	일과 중복 레이블이 있습니다. 상표는 루틴에서 고유해야합니다.
<LANGUAGE MISMATCH>	기존 루틴 코드를 컴파일 삽입할 때, 현재의 언어 모드는 일과 언어와 다릅니다
<LICENSE ALLOCATION EXCEEDED>	이 구성은 사용 가능한 모든 장치의 수영장에서 할당된 라이센스 단위 수를 초과했습니다.
<LICENSE LIMIT EXCEEDED>	현재 라이센스가 허용되는 프로세스 수를 넘으려했습니다
<LICENSE SERVER UNAVAILABLE>	현재 라이센스 서버에 연결할 수 없습니다. 네트워크를 확인하십시오.
<LIST>	잘못된 형식의 목록이 사용됩니다
<LOCKLOST>	이 작업으로 한번 잠긴 것들 중 몇 가지가 재설정되었습니다
<LOGIN INHIBITED>	시스템을 초기화하고 있습니다. 사용자가 작업할 수 없습니다.
<MAGTAPE>	테이프 처리에서 오류가 발생했습니다. \$ZA 확인하십시오.
<MAXARRAY>	이 수준으로 첨자가 너무 많습니다
<MAXINCREMENT>	변수 \$INCREMENT 하려고했는데, 값은 변경되지 않았습니다
<MAXNUMBER>	연산 처리하여 구현이 허용하는 범위 이상의 숫자가 생성되었습니다

<MAX ROUTINES>	새로운 루틴을 호출에 할당된 슬롯이 없습니다
<MAXSCOPE>	31 레벨 이상의 <b>New</b> 명령을 실행하려고했습니다
<MAXSTRING>	구현 허용된 길이보다 긴 문자열을 지정하거나 생성하려고했습니다 (32,767 자)
<METHOD DOES NOT EXIST>	메서드가 지정된 클래스 또는 지정된 개체에 존재하지 않습니다
<MNEMONICSPACE>	관련 니모닉 공간이없는 장치에서 컨트롤 니모 닉을 사용하려고했습니다
<MV WRAPUP>	MultiValue 셀 종료시 내부 오류가 발생했습니다.
<NAKED>	Naked 상태가 정의되지 않은 경우, Naked 글로벌 레퍼런스를 사용하려고했습니다
<NAME>	이름에 잘못된 구문이 있습니다
<NAMEADD>	<b>Open</b> 명령으로 인해 장치의 이름 테이블 오버플로우되었습니다
<NAMESPACE>	지정된 네임 스페이스가 정의되지 않았거나 활성화되지 않습니다
<NETFORMAT>	네트워크 메시지에 오류가 있습니다. 원격 시스템이 요구 포맷에 오류를 발견했습니다. 이 심각한 오류를 해결하려면 인터 시스템즈 기술 지원 서비스에 문의하십시오.
<NETGLOREF>	네트워크 메시지에 오류가 있습니다. 원격 시스템이 요구 포맷에 오류를 발견했습니다. 이 심각한 오류를 해결하려면 인터 시스템즈 기술 지원 서비스에 문의하십시오. (이것은 DSM 시스템에서 원격 글로벌 문자열 순서로 조합되어 있거나 7 비트로 인코딩되어있는 경우에 발생할 수 있습니다)
<NETJOBMAX>	다른 고속 네트워크 프로세스를 추가할 수 없습니다. 이것은 일반적으로 전역 버퍼 수의 부족이 원인입니다.
<NETLOCK>	31 보다 큰 원격 시스템 인덱스를 사용하여 원격 컴퓨터에 Caché ObjectScript <b>Lock</b> 명령을 실행하려고했습니다. 이 문제를 해결하려면 32 보다 적은 원격 컴퓨터를 포함하는 네트워크 구성을 다시 정의해야합니다.

<NETRETRY>	처리는 즉시 다시 시도하도록 네트워크 수준에서 실패했습니다
<NETSRVFAIL>	트랜잭션 COMMIT 또는 <b>SetKill,ZKill</b> 명령 실행 관련 서버 1 개가 트랜잭션이 열려있는 동안에 다시 시작한 것을 클라이언트 시스템이 검색되었습니다 Caché
<NETVERSION>	클라이언트와 서버 시스템에서 실행되는 DDP 버전이 다르기 때문에 서로의 메시지 형식을 받을 수 없습니다
<NETWORK DATA UPDATE FAILED - BLOCKNUMBER>	원격 시스템에서 데이터베이스 범위 외부 블록을 참조하려고했기 때문에, 네트워크를 통해 전송되는 업데이트가 손실되었습니다. 시스템 관리자에게 알리십시오.
<NETWORK DATA UPDATE FAILED - CLIENT-SERVER MISMATCH>	클라이언트와 서버 사이의 비호환성 때문에 네트워크 요청이 처리되지 않았기 때문에, 네트워크를 통해 전송되는 업데이트가 손실되었습니다
<NETWORK DATA UPDATE FAILED - CLUSTERFAILED>	클러스터 구성원이 전역 버퍼 잠금을 처리하는 도중 실패하여 네트워크를 통해 전송되는 업데이트가 손실되었습니다
<NETWORK DATA UPDATE FAILED - DATABASE>	서버의 Caché 데이터베이스 손상을 감지하여 네트워크를 통해 전송되는 업데이트가 손실되었습니다. 이것은 아주 심각한 시스템 오류가 발생할 수 있기 때문에 시스템 관리자에게 알리십시오.
<NETWORK DATA UPDATE FAILED - DIRECTORY>	참조 딕렉토리가 원격 시스템에 존재하지 않는 네트워크를 통해 전송되는 업데이트가 손실되었습니다
<NETWORK DATA UPDATE FAILED - DISKHARD>	서버의 Caché 를 해결 불가능한 디스크 하드웨어 오류를 발견했기 때문에 네트워크를 통해 전송되는 업데이트가 손실되었습니다. 이것은 데이터베이스 오류가 결과 발생 가능성이 있기 때문에 시스템 관리자에게 알리십시오.
<NETWORK DATA UPDATE FAILED - FILEFULL>	네트워크 작업에서 <FILEFULL> 오류가 발생했습니다
<NETWORK DATA UPDATE FAILED -	구현 허용된 길이보다 긴 문자열을 지정하거나 생성하려고했습니다 (32,767 자)

MAXSTRING>	
<NETWORK DATA UPDATE FAILED - NETFORMAT>	원격 시스템에서 요구되는 형식에 오류를 발견했기 때문에 네트워크를 통해 전송되는 업데이트가 손실되었습니다. 이 심각한 오류를 해결하려면 인터 시스템즈 기술 지원 서비스에 문의하십시오.
<NETWORK DATA UPDATE FAILED - NETGLOREF>	원격 시스템에서 요구되는 형식에 오류를 발견했기 때문에 네트워크를 통해 전송되는 업데이트가 손실되었습니다. 이 심각한 오류를 해결하려면 인터 시스템즈 기술 지원 서비스에 문의하십시오.
<NETWORK DATA UPDATE FAILED - NETVERSION>	클라이언트와 서버 시스템에서 실행되는 ECP 버전이 다르기 때문에 서로의 메시지 형식을 받을 수 없습니다
<NETWORK DATA UPDATE FAILED - PROTECT>	네트워크 작업에서 <PROTECT> 오류가 발생했습니다
<NETWORK DATA UPDATE FAILED - STRINGSTACK>	네트워크 작업에서 <STRINGSTACK> 오류가 발생했습니다
<NETWORK DATA UPDATE FAILED - STRMISMATCH>	네트워크를 통해 대량의 문자열을 처리하는 도중 오류가 발생하여 네트워크를 통해 전송되는 업데이트가 손실되었습니다
<NETWORK DATA UPDATE FAILED - SUBSCRIPT>	네트워크 작업에서 <SUBSCRIPT> 오류가 발생했습니다
<NETWORK DATA UPDATE FAILED - SYSTEM>	서버 <SYSTEM> 오류가 발생했기 때문에 네트워크를 통해 전송되는 업데이트가 손실되었습니다. 운영 체제 또는 ECP에서 허용되지 않는 작업을 수행하려고했습니다. 또는 Caché에 오류가 발생합니다. 이 경우, 가능한 한 많은 정보를 가장 가까운 지원 센터에 보고하십시오.
<NETWORK DATA UPDATE FAILED - WIDECHAR>	네트워크 작업에서 <WIDECHAR> 오류가 발생했습니다
<NETWORK DATA	네트워크를 통해 전송되는 업데이트가 손실되었습니다. 원인을

UPDATE FAILED>	모르겠습니다. 이 심각한 오류를 해결하려면 인터 시스템즈 기술 지원 서비스에 문의하십시오.
<NETWORK UNLICENSED>	응용 프로그램에서 원격 디렉토리에 액세스하려고했지만, Caché 네트워킹 라이센스가 없습니다.
<NETWORK>	일반적으로 다음 중 하나가 발생합니다 : 네트워크 시간 초과 된 로컬 포트가 연결이 끊어진 액세스 노드가 다른 원격 서버 연결이 비활성화
<NLS TABLE>	변환 테이블에 잘못된 데이터를 사용하여 NLS 변환을 수행하려고했습니다
<NO CURRENT OBJECT>	현재 개체가 없습니다
<NO MAILBOX>	내부 프로세스 통신하는 데 필요한 리소스를 사용할 수 없습니다
<NO SOURCE>	소스 라인이 루틴 소스 전역의 일과를 찾을 수 없습니다
<NOCAATCH>	THROW 문장에서 값을 받는 호출 스택에서 CATCH 식이 없습니다.
<NODEV>	작업 간의 통신에서 쓰기 전용 디바이스에 읽기 또는 읽기 전용 장치에 쓰기를 수행하려고했습니다
<NOJOB>	<b>View</b> 명령에서 잘못된 프로세스 번호를 지정하려고했습니다. 또는 <b>Job</b> 명령에서 오류가 발생했습니다.
<NOLINE>	존재하지 않는 일과 라인을 참조하려고했습니다
<NORESTART>	응용 프로그램이나 함수를 시작할 수 없습니다.
<ROUTINE>	존재하지 않는 루틴을 참조하려고했습니다
<NOSYS>	확장 참조 또는 암시적 참조를 현재의 네트워크 구성에 연결할 수 없는 원격 시스템에 실행하려고했습니다. 이 원격 시스템 테이블에 존재하지 않거나, 그것 자체가 나타나지 않는 DSM 시스템입니다.
<NOT PRIMARY VOLUME>	볼륨 시퀀스 1은 없습니다. 볼륨 레이블이 볼륨 함수와 일치하지 않습니다.
<NOTOPEN>	장치를 열 수 없거나 오픈하지 않는 장치를 사용하려고했습니다

<NULL VALUE>	NULL 을 사용할 수없는 위치에 NULL 문자열이 있습니다
<OUT OF \$ZF HEAP SPACE>	\$ ZF 힙, 입출력 매개 변수를 지원하는 데 필요한 공간이 부족합니다. 이 매개 변수는 Caché 와 \$ ZF 함수에서 실행되는 외부 프로그램 사이에 전달됩니다.

### 시스템 오류 메시지 - P에서 T

오류 코드	설명
<PARAMETER>	사용자 계정 함수 참조 또는 <b>Do</b> 명령으로 레이블 행에 전달된 매개 변수가 레이블 라인에 선언된 임시 변수 개수를 초과했습니다
<PRIVATE METHOD>	전용 메서드를 호출하려고했지만 사용할 수없는 메서드입니다
<PRIVATE PROPERTY>	개인 속성에 액세스하려고했지만 사용할 수없는 속성입니다
<PROPERTY DOES NOT EXIST>	속성은 지정된 개체의 클래스가 없습니다
<PROTECT>	권한이없는 상태 글로벌 ( <b>Read,Write,Kill</b> )를 사용하려고했습니다. 또는 \$ View 와 \$ ZU (49)같은 메모리 변경 View 명령을 사용하려고했습니다. 또는 확장 전역 구문을 사용하여 존재하지 않는 디렉토리를 사용하려고했습니다. 또는 기타 보호 오류가 발생했습니다.
<RANGE>	비트 위치가 허용 범위를 초과합니다
<READ>	레코드를 읽을 수 없습니다
<RECOMPILE>	루틴이 다른 Caché 버전 또는 인터 시스템즈의 기존 제품에서 컴파일되었습니다. 개체 코드를 전송하는 % RIMF 를 사용하여 시스템에 로드할 수 없습니다. 소스 코드로 전송 (% RO 와 % RI 사용) 다시 컴파일합니다.
<REMOTE CLASS EDITED>	객체를 생성한 후에 클래스가 원격 시스템에서 다시 컴파일되는 원격 시스템에있는 객체를 사용하려고했습니다
<REMOTE CLASS>	객체가 생성되면 클래스가 로컬 시스템에서 다시 컴파일되지 원격

RECOMPILED>	시스템에 있는 객체를 사용하려고 했습니다
<RESJOB>	예약 작업을 종료하려고 했습니다.
<ROLLFAIL>	Caché 는 <b>TRollBack</b> 처리하는 동안 오류가 발생했습니다. 이 오류는 여러 원격 컴퓨터가 실제로 룰백을 처리했는지 Caché 가 인식하지 않는다는 것을 의미합니다.
<ROUTINELOAD>	루틴을 로드하는 동안 오류가 발생했습니다. 일과는 오브젝트 코드가 손상된 것을 나타내고 있으며, 데이터베이스의 품질이 저하될 수 있습니다. 시스템 관리자에게 문의하십시오.
<SELECT>	<b>\$ Select</b> 함수의 진정한 조건이 없습니다
<SHARED MEM HEAP>	공유 메모리의 수요는 충족되지 않습니다. 이 오류를 해결하려면 시스템에서 더 큰 힙 공간을 할당합니다.
<SLMSPAN>	첨자 수준의 매팅 경계를 넘어, 글로벌을 삭제하려고 했습니다
<STACK>	인수 스택에 여유가 없거나 잘못된 유형을 포함하고 있습니다
<STORE>	프로세스의 파티션 공간이 부족합니다. 프로세스는 파티션의 사용 가능한 공간에서 최대의 연속 블록 이상의 블록이 필요합니다. \$ S 가 사용 가능한 공간을 나타낼 수도 있지만, 조각화되어 있습니다. 지역 변수의 개수와 크기를 줄이십시오. 몇 가지 변수를 제거 공간이 넓어질 때까지 유ти리티를 사용할 수 없습니다. 이 오류는 테이프와 순차 파일에서 너무 루틴 <b>ZLOAD</b> 을 실행하여 발생합니다.
<STRINGSTACK>	식이 너무 깁니다. 1 개의 명령의 인수식이 너무 많거나, 식과 많은 긴 문자열이 포함됩니다. 식을 간단하게하십시오.
<STRMISMATCH>	네트워크보다 긴 문자를 처리하는 내부적인 오류가 있습니다
<SUBSCRIPT>	첨자가 잘못된 값을 가지는지, 글로벌 참조가 너무 깁니다. 글로벌 참조 최대 길이 자세한 내용은 "Caché 글로벌 사용법"의 "글로벌 구조"장에서 " <u>첨자의 최대 길이의 결정</u> " 섹션을 참조하십시오.
<SYNTAX>	구문 오류 (오자와 키워드의 실수 등 언어 구조 형식 오류입니다.)
<SYSTEM>	운영 체제에서 허용되지 않는 작업을 수행하려고 했는지, Caché 에 오류가 발생합니다. 이 경우, 가능한 한 많은 정보를 지원 센터로 알려주시기

	바랍니다.
<TERMINATOR>	터미널이나 장치에 터미네이터없이 이미지 모드를 읽으려고했습니다. 읽을 고정 길이 없습니다.
<TOO MANY CLASSES>	너무 많은 활성 클래스에 액세스하려고했습니다
<TOO MANY LONG STRINGS>	문자열 스택에 긴 중간 문자열이 너무 많습니다.
<TOO MANY OREFS>	너무 많은 개체를 동시에 오픈하여 생성하려고했습니다
<TOO MANY USERS OF CLASS>	너무 많은 프로세스가 특정 클래스를 동시에 사용하려고합니다 (65561 이상)
<TOO MANY USERS>	너무 많은 사용자가 시스템을 동시에 사용하려고합니다
<TOOMANYFILES>	기본 운영 체제가 파일 기술자의 범위 밖에서 실행 되었기 때문에, Caché 는 파일을 열 수 없습니다
<TRANSACTION LEVEL>	응용 프로그램에서, 아직 실행되지 않은 트랜잭션 중첩이 너무 많습니다.
<TRANSLATE>	Caché 는 변환 값이없는 입력 값을 읽습니다. 따라서 Caché NLS 유틸리티 Translation 탭에서 정의한 Default Action 을 실행합니다.
<TRANSLOST>	이 작업이 시작된 분산 트랜잭션이 서버에서 비동기식으로 롤백되었습니다

#### 시스템 오류 메시지 - U에서 Z

오류 코드	설명
<UNDEFINED>	정의되지 않은 변수에 대한 참조가 있습니다
<UNIMPLEMENTED>	구현되지 않은 함수 또는 적합 명령이나 함수에 구현되지 않은 인수를 사용하려고했습니다
<UNIMPLEMENTED DOUBLE>	이러한 맥락에서, 부동 소수점의 사용을 지원하지 않습니다.

<UNKNOWN ERROR>	예기치 않은 오류가 발생했습니다. 이 심각한 오류를 해결하려면 인터 시스템즈 기술 지원 서비스에 문의하십시오.
<UNLICENSED>	사용 가능한 라이센스 키는 입력 라이선스로 암호화 데이터베이스 만들기 등의 요청된 작업을 허용하지 않습니다.
<VALUE OUT OF RANGE>	값이 최대 범위 또는 최소 범위 밖에 있습니다
<VOLUME IS NOT FORMATTED>	볼륨이 필요한 형식은 없습니다
<VOLUME SET ALREADY CREATED>	이미 포맷되어있는 Caché 데이터베이스를 포맷하려고했습니다
<WIDE CHAR>	Caché 는 1 바이트 문자가 필요한 장소에서 멀티 바이트 문자를 읽습니다
<WRITE DEMON FAILED>	라이트 데몬을 계속할 수 없습니다. 이 심각한 오류를 해결하려면 인터 시스템즈 기술 지원 서비스에 문의하십시오.
<WRITE>	레코드 쓸 수 없습니다
<WRONG NAMESPACE>	암시 개인 네임 스페이스에서 클래스를 로드하려고합니다.
<ZTRAP>	인자가없는 <b>ZTrap</b> 명령을 실행하려고했습니다

## 섀도잉 오류 메시지

섀도잉가 발생하는 일반적인 시스템 오류 외에 다음 섀도잉 특정 오류 코드가 있습니다.

### 섀도잉 오류 메시지

오류 코드	설명
<ZFILE>	파일이 없거나 열리지 않는 등의 파일과 관련된 문제가 발생하고 있습니다. 자세한 내용은 오류 로그를 확인하십시오.
<ZMISC>	이 표 이외의 다른 유형의 오류가 발생했습니다. 자세한 내용은 <ZMISC> 코드 뒤에 있습니다.

<ZOPEN>	TCP 를 사용하여 장치 부팅 오류입니다. 샐도잉 데이터베이스 서버를 종지합니다.
<ZREAD>	샐도잉 데이터베이스 서버에서 읽는 데 실패했습니다. 서버가 꺼져 있거나 현재 실행되고 있지 않습니다.
<ZSTOP>	샐도 잉이 종료 또는 사용자에 의해 종지되었습니다.

## ANSI 표준 M 오류 메시지

### ANSI 표준 M 오류 메시지

메시지 텍스트	의미
M1	Naked 표시등이 정의되지 않습니다
M2	잘못된 \$ FNumber 코드 문자열의 조합입니다
M3	1 다음 \$ Random 인수입니다
M4	\$ Select 진정한 조건이 없습니다
M5	0 다음 행을 참조합니다
M6	정의되지 않은 지역 변수입니다
M7	정의되지 않은 전역 변수입니다
M8	정의되지 않은 특수 변수입니다
M9	0 으로 나누기입니다
M10	잘못된 패턴과 일치 범위
M11	매개 변수가 전달되지 않습니다
M12	잘못된 줄 참조입니다 (음수 오프셋)
M13	잘못된 줄 참조입니다 (행을 찾을 수 없습니다)
M14	행 레벨이 1 이 아닙니다
M15	정의되지 않은 인덱스 변수입니다

M16	허용되지 않은 인수 <b>Quit</b> 습니다.
M17	인수가있는 <b>Quit</b> 이 필요합니다
M18	고정 길이 <b>Read0</b> 이상이 없습니다
M19	트리 또는 하위 트리를 자체적으로 병합할 수 없습니다
M20	행은 임시 목록을 가지고 있어야합니다
M21	임시 목록 이름이 중복되어 있습니다
M22	데이터가 전역으로있을 때, ^ \$ <i>Global</i> 구조적 시스템 변수 이름 (SSVN)를 <b>Set</b> 또는 <b>Kill</b> 합니다
M23	존재하지 않는 작업 번호에 ^ \$ <i>Job</i> 구조적 시스템 변수 이름 (SSVN)를 <b>Set</b> 또는 <b>Kill</b> 합니다
M24	첨자있는 로컬 변수 정의 중에 데이터 알고리즘을 변경했습니다
M26	존재하지 않는 환경 (존재하지 않는 네임 스페이스)입니다
M27	다시 시작할 수없는 트랜잭션을 롤백하려고했습니다
M28	범위 밖의 파라미터를 가지는 수학 함수
M29	구현 허용되지 않은 구조화된 시스템 변수 이름 (SSVN)를 <b>Set</b> 또는 <b>Kill</b> 합니다
M30	동일한 데이터 알고리즘 다른 데이터 정렬을 사용하는 전역 변수를 참조합니다
M31	장치 컨트롤 니모닉식이 선택한 니모닉 공백이없는 디바이스에 사용되었습니다
M32	장치 컨트롤 니모닉이 관련 행이없는 사용자 정의 니모닉 공간에서 사용되었습니다
M33	지정된 루틴이 존재하는 경우 ^ \$ <i>Routine</i> 을 <b>Set</b> 또는 <b>Kill</b> 합니다
M35	장치는 니모닉 공간을 지원하지 않습니다
M36	호환되지 않는 니모닉 공간입니다
M37	NULL 문자열로 인식되는 장치에서 읽기입니다
M38	잘못된 구조적 시스템 변수 이름 (SSVN)의 첨자입니다

M39	잘못된 <b>\$ Name</b> 인수입니다
M40	<b>Job</b> 명령 실행 매개 변수 목록에서 참조 외침입니다
M41	트랜잭션의 잘못된 <b>Lock</b> 인수입니다
M42	트랜잭션의 잘못된 <b>Quit</b> 입니다
M43	잘못된 범위 값 ( <b>\$ X,\$ Y</b> )입니다
M44	트랜잭션 외부 잘못된 명령입니다
M45	잘못된 <b>GoTo</b> 참조합니다
M57	레이블이 일과 여러 번 정의되어 있습니다
M58	정식 매개 변수가 부족합니다.

# SQL 오류 메시지

오류 코드 안에는 2 개의 오류 메시지가 있는 것도 있습니다. 이 경우 (일반적으로 SQL 컴파일시), 오류 메시지는 테이블, 뷰, 다른 엔티티에 대한 명시적인 참조를 포함합니다. 2 개의 메시지를 가진 코드의 경우, 각각의 메시지 "OR (또는)"로 구분하고 있습니다.

## Note :

이 문서는 음수 오류 코드를 나열하고 있지만, JDBC 클라이언트 ODBC 클라이언트는 항상 양수를 받습니다. 예를 들면, ODBC 및 JDBC 응용 프로그램이 오류 코드 30을 반환하는 경우 다음 테이블에 오류 코드 -30 확인하십시오.

## SQL 오류 코드

오류 코드	설명
100	(더 이상) 데이터가 없습니다
0	정상적으로 처리가 완료되었습니다
-1	잘못된 SQL 문장입니다
-2	'E'의 뒤에 지수를 나타내는 숫자를 찾을 수 없습니다
-3	인용문을 닫습니다 ( ")를 찾을 수 없습니다
-4	식별자, 상수, 집계, % ALPHAUP % EXACT % MVR % SQLSTRING % SQLUPPER % STRING % UPPER , \$\$,:+,-,( NOT, EXISTS 또는 FOR 중 하나가 처음으로 몇몇 낱말이 필요합니다.
-5	ORDER 에 지정된 열 번호 SELECT 목록에 일치하지 않습니다
-6	UNION 후에 계속할 경우 ORDER 에는 이름 대신 열 번호를 지정해야합니다
-7	ORDER 열이 SELECT 목록에 없습니다
-8	DATEPART () DATENAME (), DATEADD (), DATEDIFF ()에서 잘못된 DATEPART 코드
-9	호환되지 않는 SELECT 목록 UNION 을 사용하고 있습니다
-10	하위 쿼리의 SELECT 목록은 반드시 1 개의 항목이 있어야합니다
-11	조건식이 아닌 스칼라식이 필요합니다

-12	식별자, 상수 집계 ,\$\$,:,(+,-,% ALPHAUP % EXACT % MVR % SQLSTRING % SQLUPPER % STRING 또는 % UPPER 중 하나가 맨 위에있는 단어가 필요합니다.
-13	여기에 하위 쿼리가 아닌식이 필요합니다
-14	여기서는 비교 연산자가 필요합니다
-15	NOT 다음에는 조건이 필요합니다
-16	For 문장 FOR 다음에 한정자 SOME 또는 ALL 이 필요합니다
-17	for 문의는 (후에 for 조건이 필요합니다
-18	IS (또는 IS NOT) NULL 조건은 필드에만 적용할 수 있습니다
-19	집계 함수는 WHERE 절에 사용할 수 없습니다
-20	FROM 목록 레이블 이름이 중복되어 있습니다
-22	'SELECT *'다음에 계속할 경우 ORDER 에는 번호가 아닌 열 이름을 지정해야합니다
-23	레이블이 해당 테이블에 나열되어 있지 않습니다
-24	"정렬 컬럼이 모호합니다 :"
-25	쿼리가 완료된 후 입력이 감지되었습니다
-26	FROM 절이 없습니다
-27	필드가 해당 테이블에 불명확합니다
-28	호스트 변수는 % 또는 문자 중 하나로 시작되어야합니다
-29	해당 테이블에서 필드를 찾을 수 없습니다
-30	테이블 또는 뷰가 없습니다
-31	테이블에 필드가 (찾을 수 없음 / 고유 없음)
-32	외부 조인 기호 (=* 또는 *=)가 2 개의 필드 사이에 필요
-33	테이블에 필드를 찾을 수 없습니다
-34	최적화 프로그램이 사용 가능한 조인 순서를 찾을 수 없습니다

-35	INSERT / UPDATE / DELETE 은 업데이트할 수 없는보기를 허용하지 않습니다
-36	WITH CHECK OPTION (CHECKOPTION 클래스 변수)는 업데이트할 수 없는보기를 허용하지 않습니다
-37	SQL 스칼라 / 집계 / 단항 함수는 스트림 필드에서 지원되지 않습니다
-38	테이블에 마스터 매팅이 없습니다
-39	테이블에 RowID 필드가 없습니다
-40	ODBC 이스케이프 확장이 지원되지 않습니다
-41	외부 함수 호출, '\$ \$ tag ^ routine (...)'' 형식이 될 필요가 있습니다
-42	패턴 일치의 뒤로 계속 인용 닫기 ( "")를 찾을 수 없습니다
-43	테이블은 # IMPORT 스키마 이름 목록에서 불명확합니다
-44	중복 메서드 또는 쿼리 지수입니다
-45	Caché ObjectScript 쿠에리보디에 중복 메서드를 제공합니다
-46	Caché ObjectScript 쿠에리보디 필수 메서드가 없습니다
-47	잘못된 메서드 또는 쿼리 지수입니다
-48	트리거 이벤트의 잘못된 트리거 REFERENCING 절입니다
-49	트리거를 실행하는 언어가 SQL 이외의 경우 트리거 REFERENCING 절을 지정할 수 없습니다
-50	트리거를 실행하는 언어가 SQL 그렇지 않으면 트리거는 UPDATE OF <fieldlist> 절을 지정합니다
-51	SQL 문장이 필요합니다
-52	커서가 이미 선언되어 있습니다 / 선언되지 않습니다
-53	새 값으로 상수나 변수가 필요합니다
-54	VALUES 후 (마지막 첨자가 생략된) 배열 식별자가 필요합니다
-55	잘못된 GRANT <role> TO 또는 REVOKE <role> FROM 입니다

-56	GRANT / REVOKE 동작은 이러한 유형의 객체에 해당하지 않습니다
-57	트리거를 실행하는 언어가 SQL 아니면 트리거 WHEN 절을 지정합니다
-58	트리거 UPDATE OF 필드 _ 목록 절에서 중복 필드가 검색되었습니다
-59	여러 필드를 가질 수 없습니다
-60	% ALTER, SELECT, UPDATE 와 같은 동작이 필요합니다
-61	커서를 업데이트할 수 없습니다
-62	INSERT / UPDATE 에 새 값을 추가합니다
-63	데이터 예외 - 잘못된 이스케이프 문자입니다
-64	호환되지 않는 SELECT 목록을 INSERT 사용하고 있습니다
-65	양의 정수 상수 또는 변수가 필요합니다
-66	SELECT 목록에 중복 필드가 있습니다
-67	암시 조인 (화살표 구문)은 ON 절에서는 지원되지 않습니다
-68	ON 절에서는 기존의 외부 조인 (=*, *=)을 지원하지 않습니다
-69	SET <field> = <value expression>을 WHERE CURRENT OF <cursor>에서 사용할 수 없습니다
-70	Multi - Line 필드 LIKE, include () 또는 NULL 비교에만 사용할 수 있습니다
-71	Multi - Line 필드 비교의 왼쪽 피연산자 여야합니다
-72	Multi - Line 필드가 ORDER BY 절에서 올바르지 않습니다
-73	집계 함수를 ORDER BY 절이 지원되지 않습니다
-74	select 목록 별명 중복 검색되었습니다.
-75	TRIM 함수는 FROM 앞에 trim_spec 및 / 또는 trim_char 가 필요합니다.
-76	SELECT 목록 INTO 리스트의 요소 수가 일치하지 않습니다
-77	이 JOIN 맥락에서 조건부 열 참조 수 없습니다

-78	잘못된 트랜잭션 상태입니다
-79	참조 키와 참조 키는 같은 크기 여야합니다
-80	정수가 필요합니다
-81	열 제약 조건이 필요합니다
-82	여러 테이블의 % DESCRIPTION 정의를 찾았습니다.
-83	여러 테이블의 % FILE 정의를 찾았습니다.
-84	여러 테이블의 % NUMROWS 정의를 찾았습니다.
-85	여러 테이블의 % ROUTINE 정의를 찾았습니다.
-86	잘못된 필드 정의입니다. 데이터 형식이 정의되어 있지 않습니다.
-87	잘못된 테이블 이름입니다
-88	잘못된 필드 이름입니다
-89	잘못된 인덱스 이름입니다
-90	잘못된 뷰 이름입니다
-91	트랜잭션 모드는 두 번 이상 지정할 수 없습니다.
-92	READ WRITE 이 지정된 경우 격리 수준을 READ UNCOMMITTED 할 수 없습니다
-93	DIAGNOSTICS SIZE 에 조건 수는 숫자이어야합니다.
-94	OUTER JOIN 의 사용은 지원되지 않습니다
-95	작업 테이블에서 작업이 불허가로되고 있습니다
-96	지정된 격리 수준을 지원하지 않습니다.
-97	select 목록 이름 중복을 찾았습니다.
-98	라이선스 위반
-99	권한 오류 발생
-101	이미 열려있는 커서를 열려고합니다

-102	열려 있지 않은 커서 (FETCH / CLOSE / UPDATE / DELETE /...)하는 작업을 수행하려고했습니다
-103	UPDATE 또는 DELETE 를 실행하려고했지만, 커서는 어떤 줄도 없습니다
-104	INSERT 필드를 확인하는 데 실패했습니다
-105	UPDATE 필드를 확인하는 데 실패했습니다
-106	DELETE 행을 찾을 수 없습니다.
-107	RowID 또는 필드를 기반으로 RowID 를 UPDATE 수 없습니다
-108	필요한 필드를 찾을 수 없습니다. INSERT 또는 UPDATE 를 실행할 수 없습니다
-109	UPDATE 에 지정된 행이 없습니다
-110	파일의 잠금이 중복되어 있습니다
-111	"기본값 전용"RowID 또는 필드를 기반으로 RowID 에 INSERT 수 없습니다
-112	액세스 위반
-113	% THRESHOLD 위반
-114	일치하는 행이 이미 다른 사용자에 잡겨 있습니다
-115	읽기 전용 테이블은 INSERT, UPDATE, DELETE 수 없습니다
-116	값 목록과 테이블 열수의 INSERT / UPDATE 가 일치하지 않습니다
-117	집계 함수는보기에서 지원되지 않습니다
-118	알 수없는 혹은 고유하지 않은 사용자 또는 역할입니다
-119	UNIQUE 또는 PRIMARY KEY 제약 조건이 INSERT 에 대한 고유성 검사에 실패했습니다
-120	UNIQUE 또는 PRIMARY KEY 제약 조건이 UPDATE 고유성 검사에 실패했습니다
-121	FOREIGN KEY 제약 조건은 참조 테이블에서 행의 INSERT 를 참조 확인에 실패했습니다
-122	FOREIGN KEY 제약 조건은 참조 테이블에서 행을 UPDATE 를 참조 확인에 실패했습니다
-123	FOREIGN KEY 제약 조건은 참조되는 테이블에서 행을 UPDATE 참조 확인에 실패했습니다

-124	FOREIGN KEY 제약 조건은 참조되는 테이블에서 행 DELETE 참조 확인에 실패했습니다
-125	UNIQUE 또는 PRIMARY KEY 제약 조건, 제약 조건 생성 고유 검사에 실패했습니다
-126	RESTRICT 의 REVOKE 실패했다
-127	FOREIGN KEY 제약 조건, 제약 조건의 발생 참조 일관성 검사에 실패했습니다
-129	SET OPTION 로케일 속성 값이 잘못되었습니다
-130	Before Insert 트리거에 실패했습니다
-131	After Insert 트리거에 실패했습니다
-132	Before Update 트리거에 실패했습니다
-133	After Update 트리거에 실패했습니다
-134	Before Delete 트리거에 실패했습니다
-135	After Delete 트리거에 실패했습니다
-136	뷰의 WITH CHECK OPTION 검증 INSERT 실패했습니다
-137	뷰의 WITH CHECK OPTION 검증 UPDATE 에 실패했습니다
-138	읽기 전용 필드 값을 INSERT 및 UPDATE 수 없습니다
-139	업데이트시 동시 작업이 실패했습니다. 행 버전이 다릅니다
-140	잘못된 길이의 매개 변수가 SUBSTRING 함수에 전달되었습니다
-141	잘못된 입력 값을 CONVERT 함수에 전달되었습니다
-142	View - Column 목록보기 쿼리 SELECT 의 요소 수가 일치하지 않습니다
-150	클래스 정의의 공유 잠금을 가져올 수 없습니다
-201	테이블 또는 뷰 이름이 고유하지 않습니다
-300	이 테이블의 정의는 DDL 을 사용할 수 없습니다
-304	기본값없는 NOT NULL 필드를 데이터가있는 테이블에 추가하려고했습니다
-305	테이블의 행이 NULL 열에 값을 가지는 경우에 필요한 필드를 만들려고했습니다

-306	이 이름의 열이 이미 있습니다.
-307	이 테이블에는 이미 기본 키를 정의하고 있습니다
-308	이 테이블에 이미 ID 열이 정의되어 있습니다
-309	% CONTAINS 의 왼쪽 피연산자는 % Text 인터페이스를 지원하는 속성은 없습니다
-310	외래 키가 참조하는 테이블이 존재하지 않습니다
-311	이 테이블에 이미 같은 이름의 외래 키를 정의하고 있습니다
-312	잘못된 스키마 이름입니다. 이 스키마 이름 내용은 구분 식별자를 사용해야합니다
-314	외래 키가 참조하는 키 / 열 집합이 고유하지 않습니다
-315	제약 조건 또는 키를 찾을 수 없습니다.
-316	외래 키가 참조하는 키 / 열 집합이 존재하지 않습니다
-317	제약 DROP 수 없습니다 - 1 개 이상의 외래 키 제약 조건이 UNIQUE 제약 조건을 참조하고 있습니다
-319	참조되는 테이블은 기본 키가 정의되지 않았습니다
-320	테이블을 DROP 수 없습니다 - 1 개 이상의 외래 키 제약 조건이 테이블을 참조하고 있습니다
-321	보기 DROP 수 없습니다 - 1 개 이상의 뷰를 이보기를 참조하고 있습니다
-324	이 테이블에 이미 이 이름의 인덱스가 있습니다
-325	IDKEY 인덱스이며 테이블에 데이터가 존재하기 때문에 인덱스를 끊을 수 없습니다
-333	인덱스가 정의되어 있지 않습니다
-356	SQL 함수 (저장 프로시저 함수) 값을 반환하도록 정의되어 있지 않습니다
-357	SQL 함수 (저장 프로시저 함수) 함수 프로시저로 정의되고 있지 않습니다
-358	SQL 함수 이름 (저장 프로시저 함수) 가 고유하지 않습니다
-359	SQL 함수 이름 (저장 프로시저 함수) 를 찾을 수 없습니다

-360	클래스를 찾을 수 없습니다
-361	메서드 또는 쿼리 이름이 고유하지 않습니다
-362	메서드 또는 쿼리를 찾을 수 없습니다
-363	트리거를 찾을 수 없습니다
-364	같은 EVENT, TIME ORDER 있는 트리거가 이미 정의되어 있습니다
-365	트리거 이름이 고유하지 않습니다
-366	트리거 이름과 테이블 이름의 스키마 이름이 다릅니다
-370	포함된 SQL CALL 구문은 메서드 시저에서만 사용할 수 있습니다
-371	: HVar = CALL ... 값을 반환하지 않는 프로시저를 지정하고 있습니다
-372	외부 함수 호출이 지원되지 않습니다
-373	외부 함수 호출 % 루틴을 호출하지 않습니다
-374	테이블에 데이터가 있는 경우 스트림 유형 / 스트림 타입에 필드 데이터 형식을 변환할 수 없습니다
-375	획립되어 있지 않다 세이브 포인트를 ROLLBACK 수 없습니다
-376	지원되지 않는 CAST 대상이 지정되었습니다
-400	치명적인 오류가 발생했습니다
-401	치명적인 연결 오류
-402	잘못된 사용자 이름 및 암호입니다
-405	통신 장치에서 읽을 수 없습니다
-406	서버에 쓸 수 없습니다
-407	서버 마스터에 쓸 수 없습니다
-408	서버를 시작할 수 없습니다.
-409	잘못된 서버의 기능입니다

-410	잘못된 디렉토리
-411	필드 스트림 개체가 정의되어 있지 않습니다
-412	일반적인 스트림 오류입니다
-413	호환되지 않는 클라이언트 / 서버 프로토콜입니다
-415	SQL 파일러에서 치명적인 오류가 발생했습니다
-416	CacheInfo 오류
-417	Cache 보안 오류
-421	경고 : UPDATE 문 또는 DELETE 문에 WHERE 절이 없습니다
-422	ODBC, JDBC, 또는 동적 SQL을 통해 처리된 SELECT 요청 INTO 절을 포함할 수 없습니다
-425	저장 프로시저의 요청을 처리하는 오류입니다
-426	저장 프로시저 만들기 오류
-427	잘못된 저장 프로시저 이름입니다
-428	저장 프로시저를 찾을 수 없습니다
-429	저장 프로시저의 출력 매개 변수가 올바르지 않습니다
-430	시저 컨텍스트를 초기화할 수 없습니다.
-431	저장 프로시저의 매개 변수 유형이 일치하지 않습니다
-450	사용자의 시간 초과로 인해 요청이 시간 초과되었습니다.
-451	서버 메시지를 받을 수 없습니다
-452	메시지 시퀀스 오류
-453	사용자 초기화 코드의 오류
-459	Kerberos 인증이 실패했습니다
-460	일반적인 오류
-461	통신 연결에 실패했습니다

-462	메모리 할당에 실패했습니다
-463	잘못된 열 번호입니다
-464	함수 순서 오류
-465	잘못된 문자열 또는 버퍼의 길이입니다
-466	잘못된 매개 변수 번호
-467	열 유형이 적용되지 않습니다
-468	인출 타입이 적용되지 않습니다
-469	드라이버를 사용할 수 없습니다
-470	옵션 값이 변경됩니다
-471	커서 이름이 중복되어 있습니다
-478	쿼리를 다시 컴파일되었습니다. 결과 집합이 일치하지 않습니다.
-499	버전이 일치하지 않습니다. 버전 5.0.13 이상이 필요합니다.
-500	폐지 행 개수 제한에 도달했습니다
-10050	WinSock : 네트워크가 다운되었습니다
-10051	WinSock : 네트워크에 연결할 수 없습니다
-10052	WinSock : 인터넷 연결을 끊거나 다시했습니다
-10054	WinSock : (시간 초과 또는 다시 시작하면) 피어에 의해 연결이 재설정되었습니다
-10055	WinSock : 유효한 버퍼 공간이 없습니다
-10056	WinSock : 소켓이 이미 연결되어 있습니다
-10057	WinSock : 소켓 연결되지 않습니다
-10058	WinSock : 소켓 중지되면 보낼 수 없습니다
-10060	WinSock : 연결 시간 초과
-10061	WinSock : 연결이 거부되었습니다

-10064	WinSock : 호스트가 다운되어 있습니다
-10065	WinSock : 호스트에 대한 경로가 없습니다
-10070	WinSock : 오래된 NFS 파일 핸들입니다
-10091	WinSock : 네트워크 하위 시스템을 사용할 수 없습니다
-10092	WinSock : WINSOCK DLL 버전이 적용되지 않습니다
-10093	WinSock : WSASTARTUP 아직 제대로 실행되지 않습니다
-11001	WinSock : 호스트를 찾을 수 없습니다.
-11002	WinSock : 권한이 없는 호스트를 찾을 수 없습니다

## 개체 오류 메시지

이 문서에서는 다음 표에서는 개체 오류 메시지를 보여줍니다. CSP 와 관련된 오류 코드에 대한 자세한 정보는 "*Caché Server Pages (CSP) 사용 방법*"의 "CSP 오류 주석" 섹션을 참조하십시오.

**개체 오류 코드 - 0 부터 199**

오류 코드	설명
1	볼륨이 이미 존재합니다
2	지도 블록 읽기가 실패했습니다
3	기본 볼륨의 매핑 블록 크기의 쓰기 오류
4	전역 디렉토리지도 블록을 읽을 수 없습니다
5	전역 디렉토리지도 블록을 쓸 수 없습니다
6	전역 디렉토리 블록을 쓸 수 없습니다
13	다음 볼륨을 열 수 없습니다
14	다음 볼륨 매핑 블럭 읽기에 실패했습니다
15	디렉토리 이름이 너무 깁니다
16	잘못된지도 번호
17	크기가 범위를 벗어났습니다
18	새 볼륨을 생성할 수 없습니다
19	파일이 이미 마운트되어 있습니다
20	파일이 이미 존재합니다
21	파일 생성하실 수 있습니다
22	현재 맵에 번호가 너무 작습니다
23	파일을 확장할 수 없습니다.

24	파일이 클러스터 탑재되어 있습니다
25	CFN 를 할당할 수 없습니다
26	호환되지 않는 마운트 상태 또는 데이터베이스가 존재하지 않습니다
27	시스템마네지데이터베스 클러스터 탑재할 수 없습니다
28	데이터베이스가 이동 중입니다
30	시스템이 클러스터의 일부가 아닙니다
31	마운트된 데이터베이스 모드를 변경할 수 없습니다
32	새 볼륨에 대한 장치 공간이 부족합니다
33	새 볼륨이 시스템 파일 크기 한계를 초과했습니다
34	새 볼륨에서 알 수없는 쓰기 오류
35	데이터베이스가 확장되고 있습니다
36	데이터베이스를 탑재할 수 없습니다
37	데이터베이스가 다른 위치에 마운트되어 있습니다
38	보조 볼륨 GVXTAB 에 여유 공간이 없습니다
39	볼륨이 읽기 전용입니다
40	클러스터 마운트베이스를 삭제할 수 없습니다
41	디렉터리를 찾을 수 없습니다
42	잘못된 데이터베이스 이름입니다
43	라이트 데몬은 레이블에 읽기 / 쓰기 플래그를 설정할 수 없습니다.
44	확장을 시작할 수 없습니다
45	일부 또는 모든 데이터베이스 파일이 삭제되지 않았습니다
51	알 수없는 예기치 않은 오류
52	잘못된 인수

53	대상을 열 수 없습니다
54	대상을 읽을 수 없습니다
55	대상에 쓸 수 없습니다
56	데이터베이스는 복원되어 있습니다
57	데이터베이스가 존재하지 않습니다
58	처리에 필요한 비트맵 블록이 너무 많습니다
59	새 비트맵 블록 크기를 할당할 수 없습니다
60	이를 위해 데이터베이스를 분리해야합니다
61	따라서 데이터베이스를 개별적으로 마운트해야합니다
62	전역 디렉토리 비우기해야합니다
63	cachetemp 클러스터 마운트할 수 없습니다
64	cachetemp 를 분리 수 없습니다
65	마운트베이스를 다시 초기화할 수 없습니다.
66	데이터베이스의 리소스 이름이 시스템에서 인식되지 않습니다
67	이 데이터베이스 암호화 키 파일이 설정되어 있지 않습니다
68	마운트된 데이터베이스의 수가 라이센스 제한을 초과합니다
69	미러링된 데이터베이스를 읽기 / 쓰기로 표시할 수 없습니다
70	*** 볼륨을 포맷하는 동안 오류가 발생했습니다
71	소유권이 없습니다
72	그런 raw 디스크 장치가 존재하지 않습니다
73	그런 디렉토리가 존재하지 않습니다
74	입출력 오류입니다
75	그런 장치와 주소가 존재하지 않습니다

76	파일에 대한 액세스가 거부되었습니다
77	장치 또는 자원이 사용 중입니다
78	파일이 이미 존재합니다
79	그런 장치가 존재하지 않습니다. 또는 부정하게 사용하고 있습니다.
80	파일 테이블 오버플로우합니다
81	열린 파일이 너무 많습니다
82	읽기 전용 파일 시스템입니다
83	Error code = % 1
84	감사 데이터베이스의 최대 크기를 0 설정해야합니다
86	데이터베이스의 기본 데이터는 사용할 수 없습니다
101	Top Pointer Level : # of blocks = % 1 % 2kb (% 3 % full)
102	Bottom Pointer Level : # of blocks = % 1 % 2kb (% 3 % full)
103	Pointer Level : # of blocks = % 1 % 2kb (% 3 % full)
104	Top / Bottom Pnt Level : # of blocks = % 1 % 2kb (% 3 % full)
105	Data Level : # of blocks = % 1 % 2kb (% 3 % full)
106	Total : # of blocks = % 1 % 2kb (% 3 % full)
107	Elapsed Time = % 1 seconds % 2
108	포인터 블록 % 2 를 처리하는 동안 유형 % 1 오류
109	노드 % 1 을 처리하는 도중 오류가 발생했습니다
110	하위 블록 % 1 의 오른쪽 링크 블록을 지정합니다
111	유형 1 오류. 뷰밧파가 열려 있지 않은지, 이 데이터 집합을 마운트할 수 없습니다.
112	이 수준의 첫 번째 블록입니다
113	% 1 의 왼쪽에 있는 포인터 블록

114	이 포인터 블록 손상하고 분석할 수 없습니다.
115	하위 블록이 손상되어, 분석할 수 없습니다.
116	첫 번째 노드로 예상하는 글로벌 참조 입력이 너무 깁니다
117	포인터 블록의 첫 번째 노드 - % 1 % 2 블록을 가리 킵니다. 이것은 % 3 을 가리키는 것으로 예상됩니다. % 3 는 이전 포인터 블록이 가리키는 마지막 하위 블록의 오른쪽 링크 포인터가 가리키는 블록입니다.
118	포인터 블록의 첫 번째 노드는 % 1 입니다. It does not
119	마지막 전역 참조를 수행하지 않습니다
120	오른쪽 링크 데이터를 기반으로 필요한 글로벌 참조와 동일합니다
121	% 1 이기 전에 포인터 블록의 마지막 하위 블록
122	하위 블록 % 2 를 가리 % 1
123	***** 글로벌 % 1 잘못된 *****
124	글로벌 ^ % 1 맞습니다
125	하위 블록에 % 1 블록 타입이 있습니다
126	여기에서 % 1 을 기대하고있었습니다
127	포인터 블록은 데이터 블록에 다음이다는 것을 예상하고있었습니다
128	포인터 블록은 데이터 블록에 큰 문자열이있는 것을 기대하지 않았습니다
129	그러나 데이터 블록의 형식 정보는
130	그러나 데이터 블록의 많은 문자열 계산은
131	그것이 없다는 것을 알 수 있습니다
132	그것이있는 것을 알 수 있습니다
133	하위 블록에 다음 블록의 시작 노드를 확인할 수 없습니다.
134	bInextpntlen4 의 길이는 0 이지만 링크 맞습니다

135	blnextpntlen4 의 길이는 0 이 아닌 올바른 링크가 없습니다
136	blnextpntlen4 길이는 글로벌 참조는 너무 깁니다
137	blnextpntlen4/blnextpntvalue4 로 기술되는 참조
138	블록의 마지막 노드 다음에 오는 것으로되어 있지 않습니다.
139	blnextpntlen4 의 길이가 다음 블록의 첫 번째 노드의 길이와 일치하지 않습니다
140	하위 블록의 값은 blnextpntlen4 있습니다
141	하위 블록 blnextpntoff4 값이 있습니다
142	이것은 데이터 블록은 없습니다.
143	blnextpntoff4 하지만 이것은 큰 데이터베이스 데이터 블록이 없습니다
144	(블록의 많은 문자열을 검색하는 중에 발견되었습니다)
145	큰 문자열의 데이터 블록 수는 % 1 입니다.
146	블록 유형은 다음을 지정합니다
147	많은 문자열이 여야합니다
148	많은 문자열이 될수는 없습니다
150	데이터 블록에 구문 오류가 있습니다.
151	대량의 문자열 정보에 있습니다
152	지도 블록 % 1 레이블 오류가 있습니다
153	하위 블록 % 1 맵 블록 % 2에서 할당되지 않습니다
154	데이터 블록이 블록 % 1에 저장된 긴 문자열을 가리킵니다
155	이것은지도 블록 % 1은 할당되지 않습니다.
156	포인터 블록이 비어 있습니다
157	하위 블록은 오른쪽 린쿠구로바루리화렌스이 있고, 이것은
158	다음 포인터 노드의 글로벌 참조와 일치하지 않습니다.

159	하위 블록의 마지막 노드는 조합 순서에서 이전 데이터가되어야합니다.
160	동일해야합니다
161	하위 블록의 오른쪽 링크 참조은 % 1 입니다
162	포인터 블록의 다음 참조는 % 1 입니다.
163	포인터 노드의 글로벌 레퍼런스
164	하위 블록의 첫 번째 노드와 일치하지 않습니다
165	하위 블록의 첫 번째 노드는 % 1 입니다.
166	그것은 큰 데이터베이스 데이터 블록이므로
167	첫 번째 노드 % 1 의 첫 번째 blpntlen4 바이트와 일치해야합니다
168	이것은 포인터 블록 사이의 다음 포인터 노드 % 1 과 일치하지 않습니다.
169	포인터 노드는 블록 # % 1 을 지정합니다
170	그것은이 데이터베이스의 범위를 벗어
171	포인터 블록에 % 1 에 대한 올바른 링크가 있습니다
172	디스크가 더 이상 존재하지 않습니다
173	블록 % 1 포인터 블록 타입은 없습니다 : % 2
174	상위 블록 % 1 상위 포인터 블록 타입이 없습니다 : % 2
175	하위 포인터 블록 % 1 상위 포인터 블록 종류가 있습니다 : % 2
176	Big Strings : # of blocks = % 2 % 3MB (% 4 % full) # = % 1
177	Big Strings : # of blocks = % 2 % 3kb (% 4 % full) # = % 1
178	데이터베이스를 탑재할 수 없습니다
179	예기치 않은 오류가 발생했습니다 : % 1
180	Value (report to InterSystems) = % 1
181	***이 디렉토리에 더 이상의 검사가 중지되었습니다

182	***이 전역 더 이상 검사가 중지되었습니다
183	***이 수준의 다음 포인터 블록에서 검사를 계속합니다
184	데이터베이스가 탑재되지 않습니다
185	블록의 끝에 새 노드 % 1 가 삽입되었습니다
186	이 노드가이 블록에 필요한지 고려하십시오
187	블록의 처음에 새 노드 1 가 삽입되었습니다
188	다른 블록에서 변경해야합니다
189	새 노드 % 1 로 삽입되었습니다
190	이전 노드 % 1 및 후속 노드가 섞여 있습니다
191	노드가 이미 존재합니다 (노드 % 1)
192	*** 블록에 충분한 공간이 없습니다 ***
193	... 제거되었습니다 (상위 번호의 노드 아래에 섞여 있습니다)
194	블록의 첫 번째 노드가 삭제되었습니다
195	*** 유형은 % 1 입니다 - 잘못된 유형입니다
196	*** 오프셋이 올바르지 않습니다 : % 1 % 2 더 커지고하지 않습니다.
197	Top Pointer Level : # of blocks = % 1 % 2MB (% 3 % full)
198	Bottom Pointer Level : # of blocks = % 1 % 2MB (% 3 % full)
199	Pointer Level : # of blocks = % 1 % 2MB (% 3 % full)

### 개체 오류 코드 - 200 399

오류 코드	설명
200	Top / Bottom Pnt Level : # of blocks = % 1 % 2MB (% 3 % full)
201	Data Level : # of blocks = % 1 % 2MB (% 3 % full)

202	Total : # of blocks = % 1 % 2MB (% 3 % full)
203	그러나 하위 블록에 % 1에 대한 올바른 링크가 있습니다
204	***지도 오류 :지도 블록 % 1 카운트 필드는 % 2이지만 총 개수는 % 3입니다.
205	일관성 작업을 시작할 수 없습니다.
206	무결성 검사를 중단 하시겠습니까?
207	디렉토리 검사를 중단 하시겠습니까?
208	글로벌 검사를 중단 하시겠습니까?
209	이것은 큰 문자열 블록 % 1을 가리키고 있으며, 유형은 % 2입니다
210	
211	포인터 블록 글로벌 올바르지 않습니다
212	긴 문자열을 삽입할 수 없습니다
213	2k 데이터베이스 생성은 허용되지 않습니다
214	% 1 이중 포인터가 첫 번째 포인터는 % 3을 가리키는 전역 % 2입니다
215	이중 포인터, % 2를 가리키는 전역 % 1이 있습니다
216	'^ % 1'은 합리적인 글로벌 이름이 없습니다.
250	이 데이터 블록의 다음 포인터의 저장 값은 실제의 다음 포인터 또는 blnextpntlen4과 일치하지 않고, 정확하지 않습니다.
251	블록 # % 2 노드 # % 1에 잘못된 첨자의 길이가 있습니다
252	블록의 데이터 # % 1 글로벌 디렉토리의 데이터 # % 2와 일치하지 않습니다
253	빅 문자열 블록 # % 1 블록 오프셋 # % 2가 잘못되었습니다.
254	빅 문자열 블록 # % 1은 잘못된 블록 값 # % 2 포인트되고 있습니다.
300	데이터베이스가 탑재되지 않습니다
301	데이터베이스가 손상됩니다

302	데이터베이스가 읽기 전용입니다
303	기본 볼륨이 이미 % 1에 있습니다
304	보조 볼륨이 이미 % 1에 있습니다
305	보조 볼륨에 대해 다른 위치를 선택할 필요가 있습니다
306	보조 볼륨에 대해 다른 위치를 선택할 필요가 있습니다
307	% 1을 작성했지만 마운트할 수 없습니다. 마운트 오류 % 2입니다.
308	글로벌 % 1을 찾을 수 없습니다
309	글로벌 이미 정의되어 있습니다
310	CacheTemp 을 관리자 데이터베이스가 될 수 없습니다.
311	% 1 cachetemp 로 지정할 수 없습니다
312	cachetemp 을 sfn 찾을 수 없습니다
313	% 1의 수정이 실패했습니다
314	이것은 데이터베이스 파일 % 1은 없습니다
315	데이터베이스가 만들어졌지만, 포맷되어 있지 않습니다
316	% 1을 제거할 수 없습니다
317	데이터베이스를 삭제할 수 없습니다
319	% 1 일과 블록을 다시 선언이 완료되었습니다
320	생성에 실패했습니다 : % 1
321	다음의 이유로 % 1 Keep Type 을 설정할 수 없습니다.
322	% 2 네트워킹이 작동하지 않으므로 % 1이 마운트되지 않습니다
323	클러스터 마운트 % 1 실패
324	글로벌 디렉토리가 손상되었습니다.
325	다음의 이유로 % 1의 전표 유형을 설정할 수 없습니다.

326	다음의 이유로 1 % 보호를 설정할 수 없습니다.
327	% 1 KB 버퍼 (혹은 그 이상)가 설정되어 있지 않습니다
328	다음의 이유로 % 1 의 조합을 설정할 수 없습니다.
329	^ % 1 의 데이터베이스 오류 (InterSystems 에보고됩니다)
330	블록 % 1 은 어떤 전역에서도 사용되고 있지 않습니다
331	전달된 블록의 길이가 올바르지 않습니다 : % 1
332	큰 문자열 블록입니다. Block Dump 옵션을 사용합니다
333	글로벌 % 1 은 이미 존재합니다
334	글로벌 % 1 을 만들 수 없습니다
335	글로벌 % 1 원격 권한이 없습니다
336	% 1 은 올바른 이름이 아닙니다
337	% 1 파일에 쓸 수 없습니다
338	입력 파일 % 1 을 읽을 수 없습니다. 오류 % 2 입니다
339	구성 파일 % 1 은 유효하지 않습니다
340	% 1 파일은 사용할 수 없습니다
341	블록 번호 % 1 이 데이터베이스는 너무 큩니다
342	블록 % 1 맵 블록은 없습니다
343	블록 % 1 % 2 의 올바른지도 블록은 없습니다
344	기존 데이터베이스에서 지원되지 않는 함수입니다
345	관리자 데이터베이스가 탑재 해제 수 없습니다
346	데이터베이스가 존재하는 경우이 매개 변수를 수정할 수 없습니다
347	현재 % 1MB 이하의 크기를 설정할 수 없습니다
348	이 작업은 잘못된 매개 변수입니다

349	데이터베이스는 볼륨을 추가하기 전에 존재해야합니다
350	다음과 같은 이유에서 % 1 을 탑재할 수 없습니다
351	% 1 일과 블록을 다시 선언 오류
352	지도 블록 % 1 깨쳤습니다
353	% 1 의 데이터베이스는 요청된 % 3 MB 가 아니라 % 2 MB 로 작성되었습니다
354	통신 장치가 현재 사용 중입니다
355	블록 크기 % 1 데이터베이스를 만들 수 없습니다
356	데이터베이스 % 1 은 마운트할 수 없습니다. 리소스 % 2 시스템에 알 수 없습니다
357	FileCompact 실패했습니다. 글로벌 버퍼 공간이 부족합니다
358	FileCompact 실패했습니다. 이전 버전의 큰 문자열 블록이 있습니다
359	FileCompact 실패했습니다. 데이터베이스를 확장하고 있습니다

#### 개체 오류 코드 - 400에서 599

오류 코드	설명
400	행 % 1, 줄 : '% 2 = % 3'
401	행 % 1
402	충분한 필드가 없습니다
403	잘못된 줄 것입니다. 줄 : '% 1'
404	파일 '% 1'버전을 찾을 수 없습니다
405	잘못된 버전 '% 1'입니다
406	잘못된 매개 변수 이름 '% 1'입니다
407	잘못된 속성 값 '% 1'
408	다음 매개 변수 섹션 % 1 부족합니다 : '% 2'

409	섹션 % 1 을 제거할 수 없습니다
410	필드가 너무 많습니다
411	% 1 파일은 편집되었습니다. 관리 포털에서 변경할 수 없습니다.
412	% 1 은 잘못된지도 키워드입니다
415	% 1 은 잘못된 섹션 이름 또는 중복 섹션 이름입니다
416	중복 행을 검색되었습니다
417	중복된 항목 % 1 이 검출되었습니다
418	섹션 '[% 1]'이미 존재합니다
419	% 1 % 2 가 이미 존재합니다
420	% 1 % 2 는 존재하지 않습니다
421	% 1 의 맵 % 2 네임 스페이스 % 3 존재하지 않습니다
422	% 1 의 맵 % 2 네임 스페이스 % 3 에 이미 존재합니다
423	서버 % 1 을 제거할 수 없습니다. 데이터베이스 % 2 로 사용 중입니다
424	장치 이름을 별칭과 일치하는 것이 없습니다
425	데이터 서버 % 1 이 정의되어 있지 않습니다
426	데이터 서버 % 1 은 시스템 데이터베이스는 사용할 수 없습니다
427	시스템 데이터베이스 % 1 을 제거할 수 없습니다
428	네임 스페이스 % 1 은 이미 존재합니다
429	데이터베이스 % 1 을 제거할 수 없습니다. 네임 스페이스 % 2 로 사용 중입니다
430	MountAtStartup, ClusterMountMode 및 MountRequired 는 원격 서버에서 사용할 수 없습니다
431	시스템 데이터베이스는 클러스터 마운트할 수 없습니다
432	필요한 데이터베이스 % 1 이 정의되어 있지 않습니다

433	네임 스페이스 % 1 이 존재하지 않습니다
434	시스템 네임 스페이스 % 1 을 제거할 수 없습니다
435	필요한 네임 스페이스 % 1 이 정의되어 있지 않습니다
436	[Databases 섹션 앞에 [% 1] 섹션을 정의해야합니다
437	[Namespaces 섹션 앞에 [Databases 섹션을 정의해야합니다
438	[Devices] 섹션 앞에 [DeviceSubTypes 섹션을 정의해야합니다
439	[% 1] 섹션 앞에 [Namespaces 섹션을 정의해야합니다
440	섹션이 없습니다 : % 1
441	이후 프로세스가 중단되었습니다
442	시스템이 클러스터 데이터베이스를 지원하지 않습니다
444	댓글 길이는 % 1 자 미만이어야합니다
445	댓글이 주석 문자 '% 1'중 하나로 시작되어야합니다
446	잘못된 중첩 코멘트 행 : '% 1'
447	끝없는 댓글 '% 1'을 찾았습니다
448	% 1 은 잘못된 루틴 타입입니다
449	% 2 가 이미 존재하는 경우, 일과 % 1 매핑할 수 없습니다
450	% 1 데이터 서버 % 1 은 이미 정의되어 있습니다
451	매핑 % 1 을 제거하고 매핑 % 2 를 삭제해야합니다
452	매핑 % 1 은 이미 존재합니다
453	네임 스페이스를 다시 활성화를 위해 시스템을 중지 할 수 없습니다
454	첨자 매핑 % 2 전에 전역 매핑 % 1 을 정의해야합니다
455	필요한 데이터베이스 % 1 을 마운트할 수 없습니다
456	[config] MaxServers 매개 변수 % 1 이상으로 설정해야합니다

457	잘못된 네임 스페이스 이름입니다
458	잘못된 서버 이름입니다
461	IPv6 은 지원되지 않습니다
462	잘못된 블록 크기 % 1 입니다
463	데이터베이스 % 1 은 ECP 미러 연결에 사용할 수 없습니다
464	ECP 서버 % 1 은 존재하지 않습니다
465	원격 서버 % 1 은 미러를 지원하지 않습니다
570	글로벌 % 1 제어 문자가 포함됩니다. 이 글로벌 복원이 실패할 수 있습니다. Cache 블록 형식을 사용하여 데이터를 저장하십시오. 자세한 내용은 % 2 파일을 참조하십시오.
571	% 1 데이터베이스의 복사본이 이미 실행하고 있습니다
572	클러스터에 탑재된 데이터베이스 % 1 을 복사하거나 바꿀 수 없습니다

#### 개체 오류 코드 - 600 - 799

오류 코드	설명
601	CSP 응용 프로그램
602	데이터 서버
603	데이터베이스
604	장치
605	전역 매핑
606	글로벌 복제
607	라이센스 서버
608	네임 스페이스
609	SQL 게이트웨이
610	일과 매핑

611	테이프
612	장치 하위 유형
613	이더넷 연결
614	UDP 연결
615	이더넷 장치
616	볼륨 세트 - UCI 매픽
617	그림자의 대상
618	그림자 소스
619	LAT 서비스
620	통신 포트
621	SQL 시스템 데이터 형식
622	SQL 사용자 데이터 형식
623	SLM 복제
624	SLM
625	저널 기록
626	원격 볼륨 세트
627	네임 스페이스
628	데이터베이스
629	장치
630	구성
631	프로젝션 타입
632	Java 응용 프로그램
633	EJB 응용 프로그램

634	C + + 응용 프로그램
635	클래스 매팅
641	% 1 '% 2'이 구성에서는 정의되지 않습니다.
642	% 1 '% 2'% 3에서 참조됩니다.
643	% 1 '% 2'는 이미 존재합니다.
644	매개 변수 '% 1'이 잘못되었습니다 : '% 2'
645	% 1 '% 2'가 존재하지 않습니다.
646	변경하기 위해 다시 시작할 필요하기 때문에, 조직은 다시 활성화할 수 없습니다.
647	구성 % 1 로드 오류 : % 2
648	Activate () 메서드를 호출하기 전에 구성 % 1 % Saved ()해야합니다
649	구성 % 1은 다른 프로세스에 의해 사용 중입니다
650	% 1 네임 스페이스 '% 2'에 이미 정의되어 있습니다.
651	시작 구성은 '% 1'로 설정할 수 없습니다.
652	클러스터된 구성에서 비어 있지 않은 PIJDirectory 가 필요합니다
653	서브 스크립트 참조는 '~'문자를 포함할 수 없습니다.
654	첨자 참조는 시작 괄호로 시작되어야합니다
655	첨자 참조 종료 괄호로 끝나야합니다
656	시작 괄호는 종료 괄호 앞에 있습니다
657	참조 % 1 첨자 # 1 첨자가 잘못되었습니다
658	참조 % 1 첨자 # % 2의 첨자가 잘못되었습니다
659	잘못된 범위 지정합니다
660	지정된 범위에 3 개 이상 참조가 있습니다
661	Config API의 설정에 이름이 필요합니다

662	키가 필요합니다
663	구성 설정에 대한 정보를 찾을 수 없습니다 : % 1
664	속성이 존재하지 않음
665	구성 개체를 열 수 없습니다 : % 1
666	원격 시스템의 상태를 변경하지 못했습니다.
667	구성 파일의 구문 분석 오류 : % 1
668	다시 활성화 오류 : % 1
669	입력된 데이터 # % 1 글로벌 디렉토리에있는 ^ % 3 일치 # % 2 와 일치하지 않습니다
701	LDAP 오류 (% 1) % 2
702	LDAP 또는 전달된 인수가 초기화되지 않았습니다
703	LDAP 공유 라이브러리 (% 1)로드하지 못했습니다
704	값이 32K 의 경계에 도달했습니다
705	LDAP 힙에서 필요한 공간을 얻을 수 없습니다.
706	잘못된 부모입니다
707	예기치 않은 개체가 전달되었습니다
708	LDAP 예기치 않은 라이브러리 버전 - 기대하고 있던 것은 % 1 로드된 % 2 입니다
709	서버는 다른 사용자 확인을위한 질문을 전달했습니다. 질문에 대한 응답을 확인하고 SASLConnect 를 다시 호출하여 응답을 보냅니다
710	잘못된 매개 변수가 전달되었습니다
711	요청이 지원되지 않습니다
712	지정된 SASL 메커니즘이 지원되지 않습니다
725	클라이언트 유형의 상대 인증서 인증 수준이 올바르지 않습니다
726	확장 CihperSuite 목록에 값이 포함되어 있지 않습니다

727	SSL 통신은 현재 라이센스는 사용할 수 없습니다
-----	-----------------------------

### 개체 오류 코드 - 800 - 999

오류 코드	설명
800	서비스 % 1 로그인이 잘못되었습니다
801	로그인이 잘못되었습니다
802	서비스 % 1 로그인이 잘못되었습니다. 시스템이 실행 중입니다.
803	로그인이 잘못되었습니다. 시스템 종료 중입니다.
804	Kerberos 로그인 서비스 % 1에서 허용되지 않습니다
805	Kerberos 데이터 무결성 로그인 서비스 % 1에서 허용되지 않습니다
806	Kerberos 데이터 암호화 로그인 서비스 % 1에서 허용되지 않습니다
807	O / S 로그인 서비스 % 1에서 허용되지 않습니다
808	Kerberos 로그인이 서비스 % 1 이 필요합니다
809	서비스 % 1 이 존재하지 않습니다
810	잘못된 사용자 이름이나 암호입니다
811	Kerberos K5CCache 로그인 서비스 % 1에서 허용되지 않습니다
812	Kerberos K5Prompt 로그인 서비스 % 1에서 허용되지 않습니다
813	Kerberos K5API 로그인 서비스 % 1에서 허용되지 않습니다
814	Kerberos K5KeyTab 로그인 서비스 % 1에서 허용되지 않습니다
815	사용자는 서비스 % 1에서 인증되지 않았습니다
816	잘못된 인증 옵션 % 1 입니다
817	클라이언트 IP 주소 % 1 % 2 서비스에서 인증되지 않았습니다
818	서비스 % 1 를 삭제할 수 없습니다

819	서비스 % 1 은 이미 존재합니다
820	서비스 % 2 잘못된 인증 옵션 % 1 입니다
821	액세스가 거부되었습니다 : % 1 에 액세스할 수 없습니다
822	액세스가 거부되었습니다
824	사용자 이름 또는 암호가 잘못되었습니다
825	SQL 을 초기화할 수 없습니다 % 1
826	ZSTART 을 실행할 수 없습니다 % 1
827	사용자 % 1 은 인증되지 않습니다
828	사용자 % 1 의 계정을 중지합니다
829	사용자 % 1 를 % 2 를 추가할 수 없습니다
830	사용자 % 1 마지막 로그인을 업데이트할 수 없습니다
831	사용자 % 1 이름 또는 암호가 잘못되었습니다
832	사용자 % 1 암호 업데이트 오류
833	로그인 시간 초과
834	로그인이 중단되었습니다
835	사용자 % 1 면 시스템 보안을 우회하고 있습니다
836	프로그래머의 액세스 권한이 부족합니다
837	사용자 % 1 은 이미 존재합니다
838	사용자 % 1 은 존재하지 않습니다
839	수퍼유저 % 1 을 제거할 수 없습니다
840	% 1 을 제거할 수 없습니다. % All 역할을 가진 사용자만이 삭제할 수 있습니다.
841	기본 사용자 % 1 을 제거할 수 없습니다
842	사용자 이름 % 1 이 잘못되었습니다

843	사용자 이름 % 1 % 2 서비스가 사용 중입니다
844	네임 스페이스 % 1 데이터베이스 % 2 % 3 리소스에 대한 권한이 부족합니다
845	암호 길이 또는 패턴의 요구 사항과 일치하지 않습니다
846	사용자 이름에 도메인 이름을 포함할 수 없습니다
848	시스템 보안 구성 % 1 은 이미 존재합니다
849	시스템 보안 구성 % 1 이 존재하지 않습니다
850	감사 데이터베이스 % 1 은 사용할 수 없습니다
851	잘못된 감사 이벤트 이름 : % 1 입니다
852	감사 이벤트 % 1 은 이미 존재합니다
853	감사 이벤트 % 1 이 존재하지 않습니다
854	시스템 감사 이벤트 % 1 을 제거할 수 없습니다
855	시스템 감사 이벤트 % 1 을 변경할 수 없습니다
856	% 1 에 대한 감사 중지 오류
857	% 1 에 대한 감사를 시작할 수 없습니다.
858	감사 파일을 지우려면 시스템을 종료할 수 없습니다
859	감사 레코드 % 1 이 존재하지 않습니다
860	% 1 의 보안 레이블을 초기화할 수 없습니다. 자원은 % 2 입니다
861	권한있는 응용 프로그램 % 1 이 잘못되었습니다.
862	사용자는 권한있는 응용 프로그램 % 2 의 실행을 제한할 수 있습니다 - 실행할 수 없습니다
863	권한있는 응용 프로그램 % 1 잠겨 있습니다
864	인증된 사용자 이름이 필요합니다
865	데이터베이스 % 2 의 루틴 % 1 응용 프로그램 % 3 에 역할을 추가하는 인증이되지

	않습니다
866	클라이언트 응용 프로그램 % 1 역할을 추가하기 위해 인증되지 않습니다 - 서명 % 2
867	권한있는 응용 프로그램 % 1 을 만들 수 없습니다 - 그 이름의 응용 프로그램이 이미 존재합니다
868	권한있는 응용 프로그램 % 1 을 찾을 수 없습니다
869	애플 리케이션 % 1 이 존재하지 않습니다
870	시스템 애플 리케이션 % 1 을 제거할 수 없습니다
874	성냥 역할 % 1 이 중복되어 있습니다
875	성냥 역할 % 1 이 존재하지 않습니다
878	대상 룰 % 1 이 중복되어 있습니다
879	대상 룰 % 1 이 존재하지 않습니다
880	역할 % 1 을 제거할 수 없습니다
881	역할 % 1 을 제거할 수 없습니다
883	역할 % 1 이 존재하지 않습니다
884	역할 % 1 은 이미 존재합니다
885	역할의 최대 수에 도달했습니다
886	역할 % 1 을 변경할 수 없습니다.
887	잘못된 역할 이름 % 1 입니다
890	시스템 리소스 % 1 을 제거할 수 없습니다
891	리소스 % 1 은 이미 존재합니다
892	리소스 % 1 이 존재하지 않습니다
893	시스템 리소스 % 1 을 변경할 수 없습니다
894	자원의 최대 수에 도달했습니다

895	리소스 % 1 이 중복되어 있습니다
896	잘못된 리소스 이름 % 1 입니다
897	리소스 이름 : % 2 에 대해 잘못된 권한 % 1 입니다
898	SSL 구성 % 1 은 이미 존재합니다
900	% 1 도메인을 삭제할 수 없습니다. 도메인은 사용 중입니다.
901	% 1 도메인이 이미 존재합니다
902	잘못된 도메인 이름 % 1 입니다
903	% 1 도메인이 존재하지 않습니다
904	사용자는 모든 % 1 도메인에 있어야하지만, 사용자 % 2 다릅니다
920	필드 '% 1'은 변경할 수 없습니다
921	프로세스 % 1 권한이 있어야합니다
922	처리 리소스 % 2 % 1 권한이 있어야합니다
923	처리 리소스 % 2 또는 % 3 % 1 권한이 있어야합니다
930	시스템 보안 매개 변수를 삭제할 수 없습니다
935	암호를 변경해야합니다.
939	개체 액세스 '% 1'에 필요한 권한이 없습니다
940	처리하기에는 충분하지 특권입니다
941	감사 헤더에 기록되지 않은 레코드가 있습니다
942	사용자 이름과 역할에 동일한 이름을 사용할 수 없습니다
943	사용자 % 1 룰이 없습니다
944	잘못된 유효 기간입니다
945	네임 스페이스 '% 1'감사 이벤트를 가져오는 것은 금지하고 있습니다
946	사용자 % 1 에 액세스할 수있는 네임 스페이스가 없습니다

947	비밀 번호 로그인 서비스 % 1에서 허용되지 않습니다
948	인식되지 않은 연결 메시지입니다
949	제한 시간 내에 메시지의 전체 헤더를 가져올 수 없습니다
950	잘못된 서비스 이름 % 1입니다
951	서비스 % 1에 대한 인증되지 않은 액세스는 비활성화되어 있습니다
952	잘못된 비밀 번호입니다
953	잘못된 기존의 암호입니다
954	잘못된 비밀 번호입니다. 기존의 암호를 변환할 수 없습니다
955	사용자 % 1에 대한 잘못된 Kerberos 사용자 이름이나 암호입니다
956	Kerberos 오류 : % 1
957	비밀 번호 로그인 응용 프로그램 % 1에서 허용되지 않습니다
958	암호 패턴이 잘못되었습니다 '% 1'
959	사용자 % 1의 계정이 만료되었습니다
960	사용자 % 1의 계정은 활성 상태가 없습니다
961	Kerberos 인증은 현재 라이센스는 사용할 수 없습니다
962	Cache Direct 클라이언트를 업그레이드해야합니다
963	이 서비스는 인증이 불가능합니다
964	LDAP 서버를 중지합니다 - % 1 % 2 % 3
965	LDAP 검색의 결합에 실패했습니다 오류 % 1 % 2
966	LDAP 검색에 실패했습니다 오류 % 1 % 2
967	LDAP 를 카운트 항목에 실패했습니다 오류 % 1 % 2
968	사용자 % 1 LDAP 데이터베이스에 존재하지 않습니다
969	사용자 % 1 LDAP 데이터베이스에서 고유하지 않습니다

970	LDAP 의 첫 번째 항목에 실패했습니다 오류 % 1 % 2
971	잘못된 LDAP 암호입니다 오류 % 1 % 2
972	사용자 % 1 LDAP 사용자가 없습니다
973	사용자 % 1 은 대체하지 않습니다
974	사용자 % 1 Cache 사용자 대신 LDAP 사용자, 대체, Kerberos 사용자 또는 O / S 유저입니다
975	LDAP Get DN 에 실패했습니다. 오류 % 1 % 2
976	LDAP Get Values Len 에 실패했습니다. 오류 % 1 % 2
977	속성값 % 1 \$ list 형식으로해야합니다
978	연결하려면 사용자가 % 1 역할을 가지고 있어야합니다
979	SSL 구성 % 1 이 존재하지 않습니다
980	SSL 구성 % 1 을 활성화할 수 없습니다
981	잘못된 SSL 구성 이름 % 1 입니다
982	지정한 암호 스위트 모두에서 서버 인증이 요구되고 인증서 파일과 개인키를 포함하는 파일이 필요합니다
983	인증서 파일이 지정된 경우에는 개인 키를 포함하는 파일이 필요합니다
984	비밀키를 포함하는 파일을 지정하면 인증서 파일이 필요합니다
985	비밀키 암호가 지정된 경우에는 개인 키를 포함하는 파일이 필요합니다
986	상대 인증이 지정된 경우 CA 파일이나 CA 경로가 필요합니다
987	SSL 구성 % 1 이 잘못되었습니다
988	SSL 핸드 셰이크가 실패했습니다
989	SSL 연결에 실패했습니다. 서버 주소와 포트 (URL 이 아님)이 지정되어 있는지 확인하십시오.
990	SSL 클라이언트 테스트에만 가능합니다

991	호스트 및 포트를 지정해야합니다
992	암호가 잘못되어 있습니다
993	LDAP 인증 사용자를 수정할 수 없습니다
994	대체 인증 사용자를 수정할 수 없습니다
995	들어오는 연결에 SSL / TLS 가 필요합니다
996	SSL / TLS 가 들어오는 연결에 대해 구성되지 않습니다
997	사용자 % 1 이 O / S 대행 인증에 실패했습니다
998	CSP 세션이 만료되었습니다
999	사용자 % 1 CSP 세션이 만료되었습니다

#### **개체 오류 코드 - 1000 - 1199**

오류 코드	설명
1000	그림자 구성 '% 1'이 불완전합니다 : 소스 IP 주소나 DNS 이름이 지정되어 있지 않습니다
1001	그림자 구성 '% 1'오류 : 소스 포트 번호가 잘못되었습니다 : % 2
1002	그림자 구성 '% 1'이 불완전합니다 : 복사된 저널 파일을 저장할 디렉토리를 지정하지 않습니다
1003	그림자 구성 '% 1'이 불완전합니다 : 시작 지점이 지정되어 있지 않습니다
1004	그림자 구성 '% 1'오류 : 관리자 디렉토리 % 2 샤도우데이터베스으로 사용할 수 없습니다
1005	그림자 구성 '% 1'이 불완전합니다 : 데이터베이스 매핑이 존재하지 않습니다
1006	그림자 ID '% 1'이 잘못되었습니다 : 문자 '~'을 사용 할 수 없습니다
1007	그림자 구성 '% 1'오류 : 복사된 저널 파일의 위치로 % 2 (현 또는 대체 전표 디렉토리)는 사용할 수 없습니다
1008	그림자 구성 '% 1'오류 : 서버에 복제 모드 % 2 클라이언트 모드 % 3 과 일치하지 않습니다

1009	클러스터 구성원으로 복제가 지원되지 않습니다
1010	중지된 쉐도우 '% 1'을 다시 시작할 수 없습니다
1012	그림자 구성 '% 1'의 설정에 대한 단독 액세스를 얻을 수 없습니다
1013	그림자 구성 ID 를 지정해야합니다
1014	그림자 구성 '% 1'은 존재하지 않습니다
1015	쉐도우 '% 1'의 테스트에 실패했습니다 : % 2
1016	쉐도우 '% 1'의 테스트 시간이 초과되었습니다.
1017	쉐도우 '% 1'을 실행하지 마십시오
1020	데이터베이스 서버와 샤토우사바의 샤토우뿌로토코루에 호환되지 않습니다 : 데이터베이스 서버 버전 '% 1'에 대해 샤토우사바 버전은 '% 2'입니다
1021	데이터베이스 서버 및 샐도 서버 전표 버전은 호환되지 않습니다. 데이터베이스 서버에서 버전 % 1 인데 반해 그림자 서버 버전 % 2 입니다
1022	연결은 데이터베이스 서버 % 1 에 의해 거부되었습니다
1023	서버에서 인식할 수없는 버전 '% 1'가 발생했습니다
1024	일반 메모리 힙에서 메모리 할당 오류가 발생했습니다 : % 1
1025	샐도 잉에 사용할 수있는 일반 메모리 힙이 있습니다
1026	서버에서 인식할 수없는 메시지 '% 1'가 발생했습니다
1027	클러스터 샐도잉 요청이 거부되었습니다. 데이터베이스 서버 % 1 이 클러스터에 참가하지 않습니다
1028	클러스터 샐도잉 요청이 거부되었습니다. % 2 로 지정되는 샐도잉 소스 클러스터 데이터베이스 서버 % 1 이 참가하고 있지 않습니다
1029	샐도 잉이 오류로 인해 중단되었습니다
1030	그림자 서버 프로세스를 종료할 수 없습니다
1031	쉐도우 '% 1'다른 프로세스가 중지됩니다

1032	쉐도우 '% 1% 2 초 이내에 일시 중지할 수 없습니다
1033	요청된 저널 파일 '% 1'은 소스에 존재하지 않습니다
1034	요청된 파일 '% 1'은 소스에서 적절한 저널 파일이 없습니다
1035	저널 파일 '% 1'은 눈물입니다
1036	% 1 파일을 열 때 오류 : % 2
1037	섀도 복사본 % 1 원본 전표 % 2 파일 앞에 있습니다
1038	저널 % 2 파일에 잘못된 주소 % 1에 있습니다
1039	섀도 잉을 시작하거나 다시 저널 파일이 지정되지 않습니다 - 처음에 지정한 저널링 파일 이름이 잘못된 가능성이 있습니다
1040	1 개의 업데이 터가 종료되었다 때문에 데이터베이스 동기화를 업데이 트하지 못했습니다.
1041	클러스터 섀도 잉의 시작 지점을 찾을 수 없습니다
1042	클러스터 섀도 잉의 시작이 완전하지 않습니다 : % 1
1043	섀도잉 현재 라이센스는 사용할 수 없습니다
1044	그림자는 이미 실행 중입니다
1045	클러스터 섀도잉 검사가 잘못되었습니다 : % 1
1046	데이터베이스 업데이 트는 현재 섀도잉 소스에서 전표되지 않습니다 - 그림자 데이터베이스가 비동기임을 나타냅니다
1047	그림자는 일시 중지하고 있지 않기 때문에, 계속 작업이 잘못되었습니다
1048	그림자 중지하고 있지 않기 때문에, 시작 작업도 재개 작업도 잘못되었습니다
1070	포트 % 2 의 % 1 에 연결 시도 시간이 초과되었습니다 - 데이터베이스 서버가 실행되고 있지 않거나 네트워크가 다운되어 있습니다
1071	TCP 읽기 시간 초과 - 원격 서버가 응답하지 않습니다
1072	데이터베이스 서버가 연결이되어 있습니다 - 서버에 % 1 을 중단되었습니다

1073	그림자 서버 (% 2)이 절단되어 있습니다 - 서버에 % 1 을 중단되었습니다
1074	일과 % 1 을 종료할 수 없습니다
1075	다른 작업 (PID % 1)가 그림자 저널 파일을 삭제 중이기 때문에 삭제를 시작할 수 없습니다.
1076	삭제는이 그림자는 사용할 수 없습니다
1077	응답 가져오기 오류 : % 1
1078	작업 (PID % 1) 종료 오류 : % 2
1079	저널 동기화하지 않기 때문에 제거가 중단되었습니다
1080	저널 파일 % 3 처리할 때 셜도베이스 % 1 마운트 오류 - 소스 데이터베이스 % 2 의 이것 이후 업데은 셜도 데이터베이스에 적용되지 않습니다
1090	소스 % 1 데이터베이스가 없거나 읽을 수 없습니다
1091	소스 % 1 의 데이터베이스는 현재 마운트되어 있지 않습니다
1092	잘못된 소스 디렉토리 % 1 입니다 - 이름이 너무 길거나, 구문이 잘못되었습니다
1093	파일 '% 2'의 오프셋 % 1 에 잘못된 저널 EOF 가 있습니다 - 전 방향으로 검색하여 최종 위치를 가져옵니다
1100	레코드 읽기 위해 저널링 파일 '% 1'을 여는 데 실패했습니다
1101	파일 '% 1'이 존재하지 않습니다
1102	파일 '% 1'은 합리적인 저널 파일이 없습니다
1103	'% 1'이전 파일 가져오는 중 오류 : % 2
1104	저널 파일 '% 1'의 인스턴스를 만드는 데 실패했습니다
1105	저널 파일 '% 1'의 첫 번째 레코드가 잘못되었습니다
1106	저널 파일 '% 1'삭제시 오류 : % 2
1107	검색 문자열이 지정되어 있지 않습니다
1108	저널 파일이 지정되어 있지 않습니다

1109	저널 파일 '% 1'은 이전 파일이 있는 것이지만, 그 파일이 존재하지 않습니다
1110	저널 파일 '% 1'은 적절한 기록이 없습니다
1111	다음 저널링 파일 '% 1' 가져오는 중 오류 : % 2
1112	저널 파일 '% 1'의 오프셋 % 2 % 3 사이가 손상되어 있습니다
1120	알 수없는 열입니다 : % 1
1121	저널 레코드에 잘못된 디렉토리가 있습니다
1122	저널 레코드에 잘못된 글로벌 노드가 있습니다
1140	저널링 시작 오류 : % 1
1141	업무 일지 중지 오류 : % 1
1142	저널 파일 전환 오류 : % 1
1143	디렉토리 '% 1'은 존재하지 않습니다
1144	디렉토리 이름 '% 1'이 잘못되었습니다
1145	디렉토리 '% 1'를 만들 때 오류 : % 2
1146	저널 파일 접두사 자식 '% 1'이 잘못되었습니다
1147	파일 이름이 '% 2YYYYMMDD.nnn' 저널 파일에 대한 디렉토리 이름 '% 1'이 너무 깁니다.
1148	쟈나루화이루빠스 ('% 1 % 2YYYYMMDD.nnn')에 쉼표를 사용할 수 없습니다.
1160	잘못된 트랜잭션 ID입니다 : % 1
1161	% 2 파일의 오프셋 % 1에서 시작된 트랜잭션 TSTART 레코드가 없습니다
1180	클러스터 저널 마커 파일을 찾을 수 없습니다
1181	클러스터 저널 마커 파일을 열 수 없습니다 : % 1
1197	시작할 때 데이터베이스 암호화 키를 사용하지 않으면 저널 암호화를 사용할 수 없습니다
1198	저널 파일을 전환하고 전표 암호화를 즉시 활성화할 수 없습니다 - 전표 파일이 현재 파일 뒤에 암호화됩니다

1199	저널 파일을 전환하고 전표 암호화를 즉시 해제할 수 없습니다 - 전표 파일이 현재 파일 후에 암호화를 중지합니다
------	--

#### 개체 오류 코드 - 1200 부터 1399

오류 코드	설명
1200	데이터베이스 암호화 키 '% 1'은 이미 유효합니다
1201	데이터베이스 암호화 키가 활성화되어 있지 않습니다
1202	'% 1'은 합리적인 데이터베이스 암호화 키 파일이 없습니다
1203	파일 '% 1'의 데이터베이스 암호화 키를 사용되는 키와 일치하지 않습니다
1204	데이터베이스 암호화 키 파일 '% 2'는 사용자 '% 1'을 찾을 수 없습니다
1205	데이터베이스 암호화 키 파일 '% 2'는 이미 사용자 '% 1'이 존재합니다
1206	데이터베이스 암호화 키를 만드는 데 실패했습니다
1207	데이터베이스 암호화 키 활성화에 실패했습니다
1208	데이터베이스 암호화 키를 비활성화할 수 없습니다 암호화 데이터베이스가 탑재되어 있습니다 : % 1
1209	잘못된 비밀 번호입니다. 적어도 문자를 % 1 포함해야합니다
1210	키 파일에서 마지막 관리자를 삭제할 수 없습니다
1211	관리자의 사용자 이름이나 암호로 와이드 Unicode 문자는 지원되지 않습니다
1212	시작 데이터베이스 암호화 키를 사용을 해제하는 것은, % 1 은 허용되지 않습니다
1213	암호화 저널 파일 '% 1'충돌 복구에 필요하기 때문에 시작시 데이터베이스 암호화 키 사용을 해제하는 것은 허용되지 않습니다
1214	데이터베이스 암호화 키를 비활성화은 % 1 의 경우, 허용되지 않습니다
1215	암호화 저널 파일 '% 1'오픈 트랜잭션도 포함되므로 데이터베이스 암호화 키 사용을 해제하는 것은 허용되지 않습니다
1216	시작 데이터베이스 암호화 키를 사용은 계속 유효합니다

1217	시작 데이터베이스 암호화 키를 사용을 해제할 수 없습니다. 암호화 데이터베이스를 시작할 때 필요합니다 : % 1
1218	시작할 때 데이터베이스 암호화 키를 사용하지 않으면 감사 암호화를 사용할 수 없습니다
1219	데이터베이스 암호화 키 랙은 실패했습니다 : 잘못된 암호 가능성

#### 개체 오류 코드 - 1400 부터 1599

오류 코드	설명
1400	사용자 % 1 은 Kerberos 사용자가 없습니다
1401	ZAUTHENTICATE 루틴 "KERBEROSAUTHORIZATION"태그를 찾을 수 없습니다. SAMPLES 네임 스페이스 ZAUTHENTICATE 일과를 확인하십시오.
1402	ZAUTHENTICATE 루틴이 없습니다. SAMPLES 네임 스페이스 ZAUTHENTICATE 일과를 확인하십시오.
1403	ZAUTHENTICATE 루틴은 ServiceName, Namespace, Username, Password 및 Properties 매개 변수가 필요합니다. SAMPLES 네임 스페이스 ZAUTHENTICATE 일과를 확인하십시오.
1404	Kerberos 인증 사용자를 수정할 수 없습니다
1405	사용자 % 1 O / S 사용자가 없습니다
1406	ZAUTHENTICATE 루틴 "OSAUTHORIZATION"태그를 찾을 수 없습니다. SAMPLES 네임 스페이스 ZAUTHENTICATE 루틴을 확인하십시오
1407	O / S 인증 사용자를 수정할 수 없습니다
1409	사용자 '% 1'은 2 요소 인증이 구성되어 있지 않습니다
1410	2 단계 인증에 잘못된 기능 코드 '% 1'입니다
1411	2 단계 인증 제한
1412	2 단계 인증에 잘못된 토큰을 받았습니다
1413	휴대 전화 서비스 공급자 '% 1'은 이미 존재합니다.
1414	휴대 전화 서비스 공급자 '% 1'은 존재하지 않습니다

1415	사용자 '% 1' 휴대폰 번호 '% 2' 가 잘못되었습니다
1416	사용자 '% 1' 휴대 전화 공급자 '% 2' 가 잘못되었습니다
1417	2 단계 인증 구성이 잘못되었습니다
1418	사용자 % 1 의 계정이 잘못된 로그인 제한에 도달했습니다
1421	사용자 '% 1' 휴대 전화 번호가 있지만 서비스 공급자가 없습니다

#### 개체 오류 코드 - 2000 2199

오류 코드	설명
2000	거울 저널 로그 (% 3) 데이터베이스 '% 2' 저널 파일 # % 1 을 찾을 수 없습니다
2001	저널 파일 '% 1' 의 헤더를 읽을 수 없습니다
2002	미러 이름이 지정되어 있지 않습니다
2003	거울 저널 로그 파일 '% 1' 을 찾을 수 없습니다
2004	미러 '% 1' 저널 로그를 열 수 없습니다
2005	미러 '% 1' 저널 로그를 읽을 수 없습니다
2006	미러 세트의 이름을 변경할 수 없습니다
2007	미러 세트에 연결된 GUID 를 변경할 수 없습니다
2008	업데이트된 복구 매개 변수를 거울 멤버로 보낼 수 없습니다
2009	미러 세트 GUID 를 정의하지 않습니다. % 1 섹션은 로드할 수 없습니다
2010	미러 구성을 로드하지 못했습니다
2011	미러 이름에 ':' 문자를 포함할 수 없습니다
2012	미러 이름이 최대 문자 길이 % 1 을 초과하는 경우
2013	거울 매개 변수는 이미 로드되어 있습니다. MirrorMember.Load ()에서 다시 로드할 수 없습니다.
2014	JoinMirror 과 ReportingGUID 을 모두 설정하지 마십시오 - % 1 을 중지하고 있습니다

2015	JoinMirror 과 ReportingGUID 을 모두 설정하지 마십시오
2016	[MirrorMember 구역에 시스템 이름이 아닙니다. 거울을 병합할 수 없습니다]
2017	[MirrorMember 섹션 미러 이름이 없습니다. 거울을 병합할 수 없습니다]
2018	[MirrorMember 섹션 미러 GUID 가 없습니다. 거울을 병합할 수 없습니다]
2019	MirrorMember.CheckSecurity 에 거울 서비스 '% 1'파일을 열 수 없습니다
2020	미러 이름이 정의되어 있지 않습니다
2021	미러 이름 '% 1'이 잘못되었습니다
2022	기본 미러 멤버 거울을 종료할 수 없습니다
2023	시스템 이름에 ':'문자를 포함할 수 없습니다
2024	시스템 이름이 최대 문자 길이 % 1 을 초과하는 경우
2025	미러 이름이 구성되어 있지 않습니다. ReportingAuthorizedIDs 를 로드할 수 없습니다.
2026	SSL DN (고유 이름) 필드는 이미 사용 중입니다
2027	SSL DN (고유 이름) 필드는 NULL 이 될 수 없습니다
2028	% 1 에 필요한 매개 변수가 없습니다. 중지합니다
2029	비동기 멤버 구성이 정의되어 있지 않습니다
2030	미러 이름이 업데이트되지 않습니다
2031	쿼리 목록 미러 % 1 이 없습니다
2032	미러 세트 이름 % 1 은 존재하지 않습니다
2033	미러 구성이 잘못되었습니다. 시스템 이름 '% 1'은 고유하지 않습니다
2034	미러 % 1 구성 할당에 실패했습니다
2035	% 1 에서 로컬로 미러 이름 또는 GUID 를 중복이 있습니다
2036	'% 1'미러 구성을 로드할 수 없습니다
2037	% 2 (% 3) % 1 미러 구성을 가져올 수 없습니다

2038	거울 멤버 이름에 ':'문자를 포함할 수 없습니다
2039	거울 멤버 이름이 최대 문자 길이 % 1 을 초과하는 경우
2040	미러 이름이 정의되어 있지 않습니다
2041	모든 미러 멤버로드]는 이미 실행되고 있습니다. 다시 수행할 수 없습니다
2042	미러링 미러 이름 (% 1)를 찾을 수 없습니다
2043	비동기 멤버 연결할 때 새 구성원을 추가할 수 없습니다.
2044	% 1 에 대한 인수는 개체가 없습니다
2045	미러 세트 구성원 % 1 을 추가할 수 없습니다
2046	미러 구성이 잘못되었습니다. 시스템 % 2 GUID (% 1)가 고유하지 않습니다
2047	기본 디렉토리 '% 1'이 잘못되었습니다. '% 2'가 필요합니다
2048	미러 세트 구성원 % 1 (# % 2)를 추가할 수 없습니다.
2049	미러링 시작하기에는 충분하지 특권입니다
2050	미러 구성이로드되지 않습니다
2051	미러 세트 이름 '% 1'이 구성되어 있지 않습니다
2052	거울 관리자 데몬 % 1 을 실행할 수 없습니다
2053	거울 저널 로그 파일 '% 1'을 만들 수 없습니다
2054	거울 저널 로그 파일 '% 1'파일을 삭제할 수 없습니다
2055	거울 저널 로그 파일 '% 1'파일을 삭제할 수 없습니다
2056	거울 가상 IP 가 유효한 주소 '% 1'이 아닙니다
2057	거울 가상 IP 인터페이스는 없습니다 '% 1'
2058	미러 데이터베이스 이름이 필요합니다이 입력되어 있지 않습니다
2059	미러 데이터베이스 이름이 최대 문자 길이 % 1 을 초과하는 경우
2060	미러 데이터베이스 이름 ':'문자를 포함할 수 없습니다

2061	미러 데이터베이스 이름 '% 1'미러 구성원 % 2에서 고유하지 않습니다
2062	다른 시스템의 미러 데이터베이스 이름의 중복을 확인할 수 없습니다.
2063	데이터베이스 '% 1'은 이미 미러링되어 있습니다
2064	데이터베이스 '% 1'은 현재 미러링되지 않기 때문에 삭제할 수 없습니다
2065	새 미러 : % 1 을 만들 수 없습니다
2066	미러링 서비스가 필요하지만 사용할 수 없습니다
2067	SSL 구성 % 1 이 필요하지만 찾을 수 없습니다.
2068	SSL 구성 % 1 은 유효하지 않습니다
2069	SSL 서버 구성 '% 1'와 클라이언트 구성 '% 2'의 이름이 다릅니다
2070	거울 가상 IP '% 1'은 유효한 주소 '% 1'이 아닙니다
2071	'% 1'미러 집합 정보를 검색하는 동안 오류가 발생했습니다. 오류 정보 : % 2
2072	다른 시스템의 문자 크기가 로컬 시스템과 다릅니다
2073	미러 데이터베이스 '% 1'이 시스템에서 찾을 수 없습니다
2074	이 시스템은 미러 멤버로 구성되어 있지 않습니다
2075	이 시스템에 정의된 다른 장애 멤버가 없습니다
2076	'% 1'미러 회원 정보를 검색하는 동안 오류가 발생했습니다. 오류 정보 : % 2
2077	이 멤버를 강제로 기본으로하는 데 실패했습니다. 이유 : % 1
2078	미러링되는 DB 는 이미 활성화되어 있습니다
2079	미러링되는 DB 를 활성화할 수 없습니다. 이유 : % 1
2080	미러링되는 DB 를 제거할 수 없습니다. 이유 : % 1
2081	이것은 장애 미러 멤버가 없습니다
2082	미러 주 노드에 연결할 수 없습니다
2083	인스턴스 이름을 찾을 수 없습니다. 이유 : % 1

2084	기존의 거울을 결합할 수 없습니다 : % 1
2085	가상 IP 에 CIDR 서브넷 마스크가 포함되지 않는가 잘못된 CIDR 서브넷 마스크가 있습니다 : % 1

#### 개체 오류 코드 - 5000 ~ 5199

오류 코드	설명
5001	% 1
5002	캐시 오류 : % 1
5003	구현되어 있지 않습니다
5004	UUID 를 생성할 수 없습니다.
5005	파일 '% 1'을 열 수 없습니다
5006	이름 '% 1'이 올바르지 않습니다
5007	디렉토리 이름 '% 1'이 잘못되었습니다
5008	파일 이름이 필요합니다
5009	디렉토리 이름이 필요합니다
5010	파일 '% 1'이 이미 열려 있습니다
5011	파일 '% 1'이 열려 있지 않습니다
5012	% 1 파일이 존재하지 않습니다
5013	형식 라이브러리를 생성할 수 없습니다.
5014	% 1 은이 버전에서는 지원되지 않습니다
5015	네임 스페이스 '% 1'이 존재하지 않습니다
5017	오류가 너무 많습니다
5018	루틴 '% 1'은 존재하지 않습니다
5019	파일 '% 1'파일을 삭제할 수 없습니다.

5020	파일 '% 1'의 이름을 변경할 수 없습니다.
5021	디렉토리 '% 1'은 존재하지 않습니다
5022	예상되는 데이터가 없습니다
5023	Java 게이트웨이 오류 : % 1
5024	파일 '% 1' '% 2'에 복사할 수 없습니다.
5025	연결 이름이 잘못되었습니다 : '% 1'
5026	잘못된 ECP 클라이언트 작업 유형입니다 : % 1
5027	파일 '% 1'은 이미 존재합니다.
5028	잘못된 루틴 명의입니다
5029	프로세스 % 1 을 제거할 수 없습니다
5030	클래스 % 1 을 컴파일하는 동안 오류가 발생했습니다
5031	일과 % 1 JOB 명령을 실행할 수 없습니다
5032	디렉토리 '% 1'을 만들 수 없습니다
5051	클래스 '% 1'이 이미 존재합니다
5052	이름이 중복됩니다 : % 1
5053	클래스 이름 '% 1'은 잘못된
5054	메서드 이름 '% 1'은 잘못된
5055	매개 변수 이름 '% 1'은 잘못된
5056	속성 이름 '% 1'은 잘못된
5057	스토리지 이름 '% 1'은 잘못된
5058	트리거 이름 '% 1'은 잘못된
5059	메서드 이름이 중복됩니다 : % 1
5060	매개 변수 이름이 중복됩니다 : % 1

5061	속성 이름이 중복됩니다 : % 1
5062	저장소 이름이 중복됩니다 : % 1
5063	트리거 이름이 중복됩니다 : % 1
5064	키 이름 '% 1'이 올바르지 않습니다
5065	키 이름이 중복됩니다 : % 1
5066	인덱스 이름 '% 1'이 올바르지 않습니다
5067	인덱스 이름이 중복됩니다 : % 1
5068	쿼리 이름 '% 1'이 올바르지 않습니다
5069	쿼리 이름이 중복됩니다 : % 1
5070	클래스 이름이 중복됩니다 : % 1
5071	제약 조건 이름이 중복됩니다 : '% 1'
5072	제약 SQL 이름이 중복됩니다 : '% 1'
5073	XML 맵 이름이 중복됩니다 : % 1
5074	XML 맵 이름 '% 1'이 올바르지 않습니다
5075	클래스 사전의 유효 기간이 만료되었습니다. 업그레이드 유틸리티 \$ system.OBJ.Upgrade ()를 실행하십시오.
5076	키 이름 '% 1'에서 '% 2'문자보다 길어지고 있습니다.
5077	인덱스 이름 '% 1'에서 '% 2'자 이상입니다.
5078	메서드 이름 '% 1'에서 '% 2'자 이상입니다
5079	속성 이름 '% 1'에서 '% 2'자 이상입니다
5080	매개 변수 이름 '% 1'에서 '% 2'자 이상입니다
5081	쿼리 이름 '% 1'에서 '% 2'자 이상입니다
5082	스토리지 이름 '% 1'에서 '% 2'자 이상입니다

5083	저장 프로시저 이름 '% 1'은 고유하지 않습니다. '% 2'에서 투영되어 있습니다.
5084	패키지 이름 '% 1'이 올바르지 않습니다
5085	패키지 이름 '% 1'은 '% 2' 문자보다 길다.
5086	메서드 구현 > 32k
5087	프로젝션 클래스 형식이 '% 1 : % 2' 필요합니다
5088	'% 1 : % 2' 프로젝션 클래스 정의가 존재하지 않습니다
5089	'% 1 : % 2'로 정의되는 프로젝션 클래스는 % Projection.AbstractProjection 의 서브 클래스가 없습니다
5090	프로젝션 '% 1 : % 2'를 만드는 동안 오류가 발생했습니다
5091	프로젝션 '% 1 : % 2'를 삭제하는 동안 오류가 발생했습니다
5092	클래스 '% 1'의 이름이 다른 사람과 중복되어 있습니다. 클래스 '% 2'는 같은 이름의 대 / 소문자 다릅니다.
5093	클래스 '% 1'의 이름이 다른 사람과 중복됩니다. 패키지 '% 2'는 같은 이름의 대 / 소문자 다릅니다.
5094	클래스 '% 1'의 멤버 이름 '% 2'와 '% 3'가 중복되어 있습니다.
5095	클래스 '% 1'의 이름이 중복되어 있습니다. 클래스 '% 2' 클래스 디스크립터에서 중복됩니다.
5096	클래스 이름 '% 1' '% 2' 자 이상입니다.
5097	속성 '% 1'의 데이터가 잘못되었습니다 : '% 2'
5098	제약 조건 이름 '% 1'에서 '% 2' 자 이상입니다
5099	프로젝트 '% 1'과 이름이 충돌합니다. 프로젝트 이름 '% 2'으로 저장하려고 하지만 그것은 동명 대 소문자 다릅니다.
5101	클래스 이름이 필요합니다
5102	환경 키워드가 필요합니다
5103	메서드 이름이 필요합니다

5104	매개 변수 이름이 필요합니다
5105	속성 이름이 필요합니다
5106	스토리지 키워드가 필요합니다
5107	저장소 이름이 필요합니다
5108	트리거 이름이 필요합니다
5109	라이브러리 이름이 필요합니다
5110	쿼리 이름이 필요합니다
5111	키 이름이 필요합니다
5112	인덱스 이름이 필요합니다
5113	XML 맵 이름이 필요합니다
5114	패키지 이름이 필요합니다
5115	데이터베이스 '% 1' 클래스디코ショ나리바존 번호가 너무 높습니다.
5116	'% 1' 클래스디코ショ나리 이전 버전입니다. 업그레이드 유틸리티 \$ system.OBJ.Upgrade()를 실행하십시오
5117	클래스 '% 1'에서 요소 유형 '% 2' 요소 '% 3'와 '% 4'는 대 / 소문자가 다른 동명입니다.
5118	클래스 '% 1' 스키마 이름이 충돌하고 있습니다. 패키지 '% 2' 가 같은 스키마를 가지고 있습니다만, 다른 이름입니다.
5119	클래스 이름 '% 1'에서 기본 resultset 패키지 이름 '% 2' 와 충돌합니다.
5122	클래스 '% 1' 인덱스 '% 2': SQLNAME '% 3' 가 고유하지 않습니다
5149	'% 3' % 1 키워드 '% 2' 유형이 잘못되었습니다
5150	'% 3' % 1 키워드 '% 2' 값이 잘못되었습니다
5151	클래스 속성 태그 '% 1' 이 올바르지 않습니다
5152	환경 키워드 '% 1' 이 올바르지 않습니다

5153	메서드 특성 키워드 '% 1'이 올바르지 않습니다
5154	매개 변수 속성 키워드 '% 1'이 올바르지 않습니다
5155	속성 특성 키워드 '% 1'이 올바르지 않습니다
5156	트리거 특성 키워드 '% 1'이 올바르지 않습니다
5157	클래스 키워드 유형 '% 1'이 올바르지 않습니다
5158	메서드 키워드 유형 '% 1'이 올바르지 않습니다
5159	매개 변수 키워드 유형 '% 1'이 올바르지 않습니다
5160	뿌로빠티키와도타이뿌 '% 1'이 올바르지 않습니다
5161	토리가키와도타이뿌 '% 1'이 올바르지 않습니다
5162	메서드 키워드 값 '% 1'이 올바르지 않습니다
5163	뿌로빠티키와도 값 '% 1'이 올바르지 않습니다
5164	키 특성 키워드 '% 1'이 올바르지 않습니다
5165	Key 키워드 유형 '% 1'이 올바르지 않습니다
5166	Key 태그 값 '% 1'이 올바르지 않습니다
5167	인덱스 특성 키워드 '% 1'이 올바르지 않습니다
5168	인덱스 키워드 유형 '% 1'이 올바르지 않습니다
5169	인덱스 키워드 값 '% 1'이 올바르지 않습니다
5170	쿼리 속성 키워드 '% 1'이 올바르지 않습니다
5171	쿠에리키와도타이뿌 '% 1'이 올바르지 않습니다
5172	ку에리키와도 값 '% 1'이 올바르지 않습니다
5173	속성 '% 1'SQL 라인은 1 보다 크고, 4096 이하 여야합니다
5174	XML 맵 속성 키워드 '% 1'이 올바르지 않습니다
5175	XML 맵 키워드 유형 '% 1'이 올바르지 않습니다

5176	클래스 키워드 값 '% 1'은 잘못된
5177	'% 2'인덱스뿌로빠티 데이터가 잘못되었습니다 : '% 1'
5178	인덱스데타뿌로빠티 '% 2'가 올바르지 않거나 transient 합니다 : '% 1'
5179	속성 '% 1'SQL 컬럼은 고유해야합니다 : '% 2'는 '% 3'로 지정되어 있습니다
5190	스트림이 InitialExpression 지원되지 않습니다 : 속성 '% 1'

#### 개체 오류 코드 - 5200에서 5399

오류 코드	설명
5201	잘못된 분석 트리입니다
5202	컴파일하는 것이 없습니다
5203	CDL 구문 분석 오류 : % 1
5251	final 메소드 '% 1'은 변경할 수 없습니다
5252	final 매개 변수 '% 1'은 변경할 수 없습니다
5253	final 속성 '% 1'은 변경할 수 없습니다
5254	final 클래스 '% 1'에서 파생될 수 없습니다
5255	final 뿌로빠티메소드 '% 1'을 무시 할 수 없습니다
5256	final behavior '% 1'을 대체할 수 없습니다
5257	키 정의 '% 1'을 무시하지 않습니다
5258	인덱스 정의 '% 1'을 재정 의할 수 없습니다
5259	쿠에리타이뿌는 '% 1'은 변경할 수 없습니다
5260	final 쿼리 '% 1'을 변경할 수 없습니다
5261	final 쿠에리메소드 '% 1'을 무시하지 않습니다
5262	매개 변수 '% 1'을 사용하여 쿼리를 뷰로 투영 수 없습니다

5263	non - SQL 쿼리 '% 1' 뷰로 투영 수 없습니다
5264	속성 % 1 SQLComputeOnChange 속성 % 2 정의되어 있지 않습니다
5265	최종 키워드 '% 1'은 변경할 수 없습니다.
5266	여러 종속 관계가 정의된 '% 1'
5267	Final XML 맵 '% 1'을 변경할 수 없습니다
5268	final XML 메서드 '% 1'은 무시하지 않습니다
5269	final 메소드 '% 1'을 무시하지 않습니다
5270	'% 1'과 '% 2'간의 중복한다하는 복합 메서드 이름이 있습니다
5271	'% 1'정의를 바꿀 수 없습니다 : '% 2'
5272	final '% 1'을 수정할 수 없습니다 : '% 2'
5273	별칭 메서드 루프가 % 1 : % 2 에서 검색되었습니다
5274	별칭 메서드 '% 3%' 1 : % 2 (% 4)에서 찾을 수 없습니다
5275	에이리아스메소드 '% 3'의 서명이 % 1 : % 2 와 일치하지 않습니다
5276	에이리아스메소드 '% 1 : % 2'가 참조하는 클래스 % 3 슈퍼 클래스가 없습니다
5277	'% 3'사부에쿠투스텐토 '% 2'에 의존 (부자) 관계 '% 1'을 도입 할 수 없습니다.
5278	'% 1'VERSIONPROPERTY 속성 '% 2'가 정의되어 있지 않습니다
5279	VERSIONPROPERTY 는 사부에쿠투스텐토 '% 1'변경할 수 없습니다
5280	계산 속성 컬렉션을 지원하지 않습니다 '% 1'(Computed 는 있지만 Calculated 는 없습니다).
5281	클래스에 여러 identity 속성을 정의합니다 : '% 1 : % 2'
5282	Identity 속성은 컬렉션에 수 없습니다 : '% 1 : % 2'
5283	Identity 속성은 정수 형식이어야합니다 : '% 1 : % 2'
5284	IDKEY 인덱스 identity 속성을 기반으로하지 않습니다 : '% 1 : % 2'

5285	속성 '% 1'SQLComputed 되고 있지만 SQLComputeCode 가 정의되어 있지 않습니다
5286	클래스 '% 2'에서 classtype '% 1'클래스 '% 4'의 '% 3'로 재정 의할 수 없습니다.
5287	클래스에 포함된 속성이 너무 많기 때문에, 인스턴스 변수가 너무 많아서 컴파일 할 수 없습니다
5288	대기열에 대기하고있는 클래스가 많은 컴파일하는 클래스를 더 이상 큐에 놓을 수 없기에, 대기열에서 대기하고있는 클래스를 컴파일 건너 뛰었습니다 클래스가 생략되었습니다 : '% 1'
5289	컴파일 트리를 만들 수 없습니다. 의존하고있는 클래스 '% 1'의 계승 관계가 해결되지 않습니다.
5301	메서드 '% 1'호출 태그가 없습니다
5302	메서드 '% 1'코드가 없습니다
5303	메서드 '% 1'표현이 없습니다
5304	메서드 '% 1'발생기가 없습니다
5305	메서드 '% 1'에 이름이 없습니다
5306	매개 변수 '% 1'에 이름이 없습니다
5307	속성 '% 1'로 이름이 없습니다
5308	검색어 '% 1'에 이름이 없습니다
5309	검색어 '% 1'유형이 없습니다
5310	SQL 뿌로시쟈메솟도 '% 1'클래스 메소드가 아니면 안됩니다
5311	SQL 뿌로시쟈메솟도 '% 1'의 컨텍스트 매개 변수가 틀립니다.
5312	제약 조건 '% 1'에 이름이 없습니다.
5313	프로젝션 '% 1'유형이 없습니다
5351	클래스 '% 1'은 존재하지 않습니다
5352	클래스 '% 1'은 최신이 없습니다

5353	클래스 '% 1'클래스 종속성을 확인할 수 없습니다.
5354	순환 상속이 발생했습니다 : % 1
5355	메서드 발생기 종속성이 미해결입니다 : % 1
5356	콘솔리ਊ스토레지쿠라스 '% 1'이 존재하지 않습니다
5357	클래스 '% 1'에 의존 미해결입니다. 이 클래스의 상위 / 하위 클래스 '% 2'미해결입니다.
5358	language = '% 1'메서드는 SQL 프로 시저로 프로젝트 수 없습니다 : '% 2'
5359	메소드제네레이터 = '% 2'에서는 언어 유형 = '% 1'은 지원되지 않습니다
5364	'% 2'에 사용되는 클래스 '% 1'이 정의되어 있지 않습니다.
5365	컬렉션 '% 1 : % 2'에서 프로젝트하는 테이블의 이름이 고유하지 않습니다 : % 3
5367	일과 배포 종속성이 미해결입니다 : % 1
5368	클래스 '% 1'개체가 % 2 프로세스 인스턴스 화합니다
5369	클래스 '% 1'프로세스 '% 2'으로 컴파일하고 있습니다
5370	메소드제네레이터 '% 1'은 존재하지 않습니다
5371	클래스 '% 1'은 공유 잠금 사용 할 수 없습니다
5372	클래스 '% 1'점유 사용 잠글 수 없습니다
5373	클래스 '% 2'가 사용하는 클래스 '% 1'은 존재하지 않습니다
5374	클래스 디스크립터 생성 중에 메서드 '% 1'에서 내부 오류가 발생했습니다. 지원 센터에 문의하십시오
5375	인스턴스 '% 1'클래스 메소드에서 사용할 수 없습니다
5376	'% 1'메서드 또는 속성이 클래스에 존재하지 않습니다.
5377	인스턴스 메서드 '% 1'클래스 메서드에서 호출하려고합니다
5378	클래스 '% 1'은 배치 모드입니다.
5379	배포 모드 클래스는 컴파 일할 수 없습니다 : '% 1'

5380	클래스 '% 1'은 배치 모드에서 '% 2'를 사용합니다.
5381	배포 모드에 있는 클래스는 내보낼 수 없습니다 : '% 1'
5382	배포 모드에 있는 클래스는 편집할 수 없습니다 : '% 1'
5383	SQL 데이터 맵만 재정 의할 수 있습니다 : '% 1'
5384	SQL Map 키워드는 최종입니다. 새 DATA 항목에만 유효합니다 : '% 1'
5385	노드 % 2 에 있는 SQL 맵 데이터 Piece 위치 % 3 은 이미 사용하고 있습니다 : '% 1'
5386	클래스 '% 2'가짜의 슈퍼 클래스에 메서드 '% 1'은 존재하지 않습니다.
5387	메서드 '% 1'클래스 '% 2'의 슈퍼 클래스에 있는 초록입니다. 호출할 수 없습니다.
5388	클래스 '% 1'이 존재하는 데이터베이스에 대한 쓰기 권한이 없습니다. 따라서 클래스 잠금을 가져올 수 없습니다
5389	메서드 '% 1'는 인스턴스 메서드 # # super 를 사용하여 클래스 '% 2'를 호출합니다. 그러나이 클래스는 '% 3'의 주요 슈퍼 클래스가 없으므로라는 것은 없습니다.
5390	클래스 '% 1'의 종속성을 확인할 수 없습니다. 조상인 상위 클래스 '% 2'가 해결되지 않습니다.
5391	클래스 '% 1'클래스 의존성이 다음 오류로 해결할 수 없습니다 : % 2
5392	이 클래스 메서드 '% 1'은 정의되지 않습니다.
5393	클래스 메서드에서 속성 '% 1'을 볼 수 없습니다.
5394	클래스 '% 1'이 의존하고 있는 클래스 '% 2' 다른 시스템 수준에 있기 때문에 먼저 컴파일되지 않습니다.
5395	클래스 '% 1'에서 레이블 '% 3'를 통해 메서드 '% 2'를 부르는 잘못된 루틴입니다.
5396	클래스 '% 1'클래스 설명자 시스템 코드에서 지원하는 크기보다 큅니다.
5397	항목 '% 1'를 포함하는 데이터베이스에 대한 쓰기 권한이 없기 때문에이 상품을 컴파 일할 수 없습니다.
5398	잠금 테이블이 가득합니다 : 단독 사용을 위한 클래스 '% 1'의 잠금 획득에 실패했습니다
5399	클래스 '% 1'을 컴파 일할 수 없습니다. 클래스 '% 2'가 최신 버전이 없습니다

## 개체 오류 코드 - 5400 부터 5599

오류 코드	설명
5400	속성을 여러 데이터 위치에 저장할 수 없습니다 : '% 1 % 2'
5401	잘못된 동작 유형입니다 : % 1
5402	잘못된 CacheDirect 맵입니다
5403	잘못된 CLIENTDATATYPE 입니다 : % 1
5404	생성기에서 잘못된 코드 모드가 발생했습니다 : % 1
5405	잘못된 컬렉션 타입입니다 : % 1
5406	잘못된 기본 스토리지 환경입니다
5407	잘못된 ID 카디입니다 : % 1
5408	잘못된 ID 카운터 : % 1
5409	잘못된 ID 종속성입니다 : % 1
5410	잘못된 ID 키 : % 1
5411	잘못된 ID 키 문자열입니다 : % 1
5412	잘못된 ID 키 속성입니다 : % 1
5413	잘못된 식별 유형입니다 : % 1
5414	잘못된 인덱스 특성 : % 1
5415	잘못된 키입니다
5416	잘못된 키 속성입니다 : % 1
5417	잘못된 메서드 코드 모드 : % 1
5418	잘못된 속성 유형입니다 : % 1
5419	잘못된 참조 타입입니다 : % 1

5420	잘못된 저장소 별칭입니다
5421	잘못된 저장소 정의입니다
5422	컨텍스트없이 사용 방법에 오류가 있습니다 : % 1
5423	데이터 매팅이 정의되어 있지 않습니다
5424	저장소 이름이 지정되어 있지 않습니다
5425	속성 매개 변수가 선언되어 있지 않습니다 : % 1
5426	속성 유형을 변경할 수 없습니다 : % 1
5427	루틴 컴파일 오류 : % 1
5428	스토리지 클래스가 지정되어 있지 않습니다
5429	저장소 '% 1'이 정의되어 있지 않습니다
5430	트리거 '% 1'이 정의되어 있지 않습니다
5431	쿼리 매개 변수가 선언되어 있지 않습니다 : % 1
5432	ROWSPEC 에 지정된 유형이 올바르지 않습니다 : % 1
5433	잘못된 ODBCTYPE 입니다 : % 1
5434	잘못된 SQLCATEGORY 입니다 : % 1
5435	잘못된 저장소 구조입니다
5436	잘못된 저장소 존입니다
5437	잘못된 저장소 리터럴 식입니다 : % 1
5438	잘못된 저장소 기호 식입니다 : % 1
5439	기호를 정의하지 않는 스토리지입니다 : % 1
5440	잘못된 시리얼 존입니다
5441	정의되지 않은 스토리지 기호입니다 : % 1
5442	데이터의 첨자는 이미 사용 중입니다 : % 1

5443	여러 ID 키가 정의되어 있습니다 : % 1
5444	여러 개의 기본 키를 정의하고 있습니다 : % 1
5445	여러 익스텐트 인덱스가 정의되어 있습니다 : % 1
5446	ID 키를 조건을 지정할 수 없습니다 : % 1
5447	기본 키에 조건을 지정할 수 없습니다 : % 1
5448	확장 색인 조건을 지정할 수 없습니다 : % 1
5449	ID 키를 사용하여 데이터를 클러스터화할 수 없습니다 : % 1
5450	확장 색인 데이터를 클러스터화할 수 없습니다 : % 1
5451	확장 인덱스 속성을 정의할 수 없습니다 : % 1
5452	확장 색인 키로 사용 할 수 없습니다 : % 1
5453	데이터 형식 클래스는 속성을 가질 수 없습니다 : % s
5454	EXTENTQUERYSPEC에 지정된 특성이 올바르지 않습니다 : % 1
5455	트리거 '% 1'이벤트가 올바르지 않습니다
5456	트리거 '% 1'이벤트가 필요합니다
5457	트리거 '% 1'시간이 잘못되었습니다
5458	트리거 '% 1'시간이 필요합니다
5459	트리거 '% 1'순서가 필요합니다
5460	트리거 '% 1'코드가 필요합니다
5461	속성 '% 1'에 대한 스트림 형식이 올바르지 않습니다
5462	'% 1'에 대한 스트림 저장 값이 올바르지 않습니다
5463	잘못된 외부 속성 : '% 1'
5464	클래스 '% 2'를 나타내는 외래 키 '% 1'이 올바르지 않습니다
5465	키 '% 2'를 나타내는 외래 키 '% 1'이 올바르지 않습니다

5466	오류 코드 '% 1'은 범위 밖에
5467	오류 이름 '% 1'이 올바르지 않습니다
5468	인덱스 '% 1'은 잘못된
5469	뷰 클래스는 속성을 가질 수 없습니다 : % s
5470	ID, 기본 키, 고유 인덱스에 의해 데이터를 재정 의할 수 없습니다 : % s
5471	비트맵 인덱스는 고유 수 없습니다 : % 1
5472	비트맵 인덱스에서 데이터를 클러스터화할 수 없습니다 : % 1
5473	제한 매개 변수가 선언되어 있지 않습니다 : % 1
5474	ID 카운터는 외부 테이블에서 잘못되었습니다 : % 1
5475	루틴 컴파일 오류 : % 1. 오류 : % 2
5476	루틴 '% 1'콘빠이루시구니챠가 올바르지 않습니다
5477	% 1 키와도시구네챠에라입니다. 키워드 '% 2'는 '% 3'이어야합니다.
5478	% 1 키와도시구네챠에라입니다. 키워드 '% 2'는 '% 3'또는 그 서브 클래스가 아니면 안됩니다.
5479	IDKEY 인덱스가 영속 클래스가 필요합니다 : % 1
5480	% 1 매개 변수가 선언되어 있지 않습니다 : % 2
5481	클래스 % 1 저장소 정의가 올바르지 않습니다
5482	클래스 % 1 저장소가 올바르지 않습니다
5483	하위 노드의 잘못된 컬렉션 타입입니다 : % 1
5484	비트맵 인덱스는 종속 클래스에서 지원되지 않습니다
5485	비트맵 인덱스는 IDKEY 가 하나의 양의 정수 특성을 기반으로하는 경우에만 지원됩니다.
5486	잘못된 메서드 언어 : % 1
5487	잘못된 ROWSPEC 형식 % 2 입니다 : % 1

5488	잘못된 % 1 formalspec 형식 % 2 입니다. % 3 이 필요합니다.
5489	뿌로빠티메솟도 '% 2 : % 3'의 메소드 생성 코드를 실행하는 동안 오류 \$ ZE = '% 1'이 발생했습니다.
5490	메서드 '% 2'의 메소드 생성 코드를 실행하는 동안 오류 \$ ZE = '% 1'이 발생했습니다.
5491	시리얼 클래스 또는 리터럴 클래스인 '% 1'과 관계를 형성할 수 없습니다
5492	리レーションシップ카ーディナリティ이 올바르지 않습니다 '% 1'
5493	리レーションシップ카ーディ나리티이 필요합니다, '% 1'
5494	반대 카디 '% 2'는 잘못된 '% 1'
5495	반대 관계가 필요 '% 1'
5496	반대 속성 '% 2'가 정의되지 않았습니다 '% 1'
5497	반대 속성 반대 '% 2'는 관계를 참조하지 않습니다 '% 1'
5498	관련 클래스 '% 2'는 컴파일되지 않습니다 '% 1'
5499	내부 관계 오류
5500	% 2 의 % 1 형식 인수 형식이 올바르지 않습니다 : % 3
5501	Cache SQL 이 설치되어 있지 않습니다
5502	SQL 테이블 '% 1'컴파일하는 동안 오류
5503	필드 이름이 잘못되었습니다 : % 1
5504	부모 열 '% 1'이 올바르지 않습니다
5505	SQL 테이블 '% 1', 부모가 올바르지 않습니다
5506	SQL 카운터 '% 1'이 올바르지 않습니다
5507	SQL 식별 테이블 '% 1'이 올바르지 않습니다
5508	맵 '% 2'SQL 맵뿌데타휘루도 '% 1'이 잘못되었습니다
5509	SQL Map 행 IDField '% 1'이 올바르지 않습니다

5510	맵 '% 2'SQL 맷뿌사부스ку리뿌토 '% 1'이 잘못되었습니다
5511	SQL Map 탑입 '% 1'이 올바르지 않습니다
5512	SQL 참조 대상 '% 1'이 올바르지 않습니다
5513	맛뿌데타휘루도 '% 1'은 합리적인 필드가 없습니다
5514	지도 식 - 알 수없는 혹은 잘못된 필드입니다 : % 1
5515	테이블 '% 1'은 이미 존재하고 있습니다
5516	테이블 '% 1'이 존재하지 않습니다
5517	테이블을 찾을 수 없습니다
5518	테이블 ID '% 1'이 존재하지 않습니다
5519	잘못된 SQL 부모 테이블입니다
5520	잘못된 테이블 참조입니다
5521	SQL 오류 : SQLCODE = % 1 % msg = % 2
5522	SQL 테이블 '% 1'을 내보낼 수 없습니다 부모를 내보낼 수 없습니다
5523	테이블 이름이 잘못되었습니다 : % 1
5524	% 2 의 {} 필드 참조가 잘못 : '% 1'
5525	'% 1'라는보기를 가지는 클래스를 찾을 수 없습니다.
5526	테이블 '% 1', '% 2'에서 참조하는 모습, 지정된하지만 존재하지 않습니다.
5527	SQL 권한 위반
5528	SQL 식별자가 잘못되었습니다 : '% 1'로 구분된 식별자 옵션이 Off로되어 있습니다.
5529	정규 SQL 식별자가 올바르지 : '% 1'은 SQL 예약어입니다. 이 % 2에 다른 SQL 이름을 지정하십시오
5530	잘못된 사용자 이름 및 암호입니다
5531	SQLMGR 클래스 이름을 찾을 수 없습니다.

5532	연결 오류
5533	할당 오류
5534	열 오류
5535	테이블 오류
5536	PrimaryKeys 오류
5537	스트림 오프셋 % 1 이동할 수 없습니다
5538	맵 '% 2'의 맵 데이터 변수 '% 1'의 표현이 없습니다
5539	맵 '% 1', 사부스쿠리뿌토레베루 '% 2'의 맵 데이터 변수 이름이 없습니다
5540	SQLCODE : % 1 메시지 : % 2
5541	맵 : % 2 -지도 식 - 알 수없는 혹은 잘못된 필드입니다 : % 1
5542	맵 : % 2 - 데이터 액세스 가능 - 잘못된 식 '% 1'입니다. 이전 첨자 수준에서 {Li}, {Di}, {iDj} 참조 여야합니다
5543	맵 : % 2 - 잘못된 조건, 다음 서브루틴 행 참조 또는 아래 첨자 중지 식 - '\$ 1'은 잘못된 식입니다. 이 첨자 수준 또는 이전 첨자 수준에서 {Li} 또는 {} Di 참조이거나 이전 첨자 수준에서 {iDj} 참조 여야합니다
5544	맵 : % 2 - 데이터 액세스 변수 식 - '% 1'은 잘못된 식입니다. 이 첨자 수준 또는 이전 첨자 수준에서 {Li}, {Di}, {iDj} 참조해야합니다.
5545	맵 : % 2 - 맵 데이터 검색 코드 - '% 1'은 잘못된 식입니다. {Li}, {Di}, {iDj} {% row} {% rowraw} 또는 {*} (이 필드) 참조해야합니다.
5546	맵 : % 2 - RowID 사양 - '% 1'은 잘못된 식입니다. {Li} 또는지도 데이터에서 어떤 필드에 있어야합니다.
5547	맵 : % 2 - 첨자 식 - '% 1'은 잘못된 식입니다. 유효한 필드 참조해야합니다. 이것이 마스터 맵이면 IDKEY 필드 여야합니다.
5548	맵 : % 2 -지도 데이터 필드 이름 - '% 1'은 잘못된 식입니다. 유효한 필드 참조해야합니다.
5549	맵 : % 2 - 맵 데이터 노드 - '% 1'은 잘못된 식입니다. {Di} 또는 {} iDj 참조해야합니다.

5550	SQL 은 클래스 % 1 메소드 % 2 에서 COS 이외의 언어에서 데이터 형식 메서드를 지원하지 않습니다.
5551	DEFAULTDATA 는 리스트 노드이어야합니다 : % 1
5552	스토리지 PARENT 토큰이 사용되고 있지만, 부모 관계가 없습니다 : % 1
5553	ID 속성의 조합은 EXACT 필요가 있습니다 : % 1
5554	% 2 매개 변수 값은 양의 정수 여야합니다 : % 1 % 2 = % 3
5555	클래스 % 1 % 2 속성 메서드 % 3 숫자 형식이 올바르지 않습니다
5556	외래 키 '% 1'의 카디널리티가 참조 키와 일치하지 않습니다
5557	BITSLICE 인덱스에 사용할 수 있는 것은 1 개의 속성만 : % 1
5558	SUBVALUE 인덱스가 정의되어 있는데, BuildValueArray 메소드가 구현되어 있지 않습니다 : % 1
5559	클래스 % 1 클래스 정의에 오류가 있기 때문에 컴파일되지 않습니다
5560	읽기 전용 메서드는 저장할 수 없습니다. 이것은 구현이 너무 커서 속성에 들어가지 않기 때문입니다
5561	인덱스에는 적어도 1 개의 속성이 필요합니다 : % 1
5562	SUBVALUE 인덱스는 고유 수 없습니다 : % 1
5563	% 2 매개 변수 값은 0 부터 15 사이의 정수이어야합니다 : % 1 % 2 = % 3
5564	스토리지 참조 : '% 2'에서 사용하는 '% 1', '% 3'로 사용하도록 이미 등록되어 있습니다
5565	'% 2'에 사용되는 참조 '% 1'등록 오류 : % 3
5566	CACHELIB 데이터베이스가 읽기 전용으로되어있는 경우, % SYS 의 모든 클래스를 재컴파일 수 없습니다
5567	클래스 '% 1'쓰기 권한이없는 데이터베이스에 있기 때문에 컴파일할 수 없습니다
5580	SQL 권한 오류 : '% 1'
5581	인덱스 작성 또는 삭제하는 중 오류 : \$ ZError = '% 1'

5582	SAMPLES 네임 스페이스 테이블, 뷰, 프로 시저에 대한 모든 권한을 _PUBLIC 에 부여할 수 없습니다 : \$ ZError = '% 1'
5583	SQL Map '% 1'데이터 필드 '% 2'노드 값 '% 3'가 잘못되었습니다. 노드 값은 인덱스 맵에서 허용되지 않습니다. 데이터 맵에서만 허용됩니다
5584	DOCBOOK 네임 스페이스 테이블 Docbook.block 의 SELECT 권한을 _PUBLIC 에 부여할 수 없습니다 : \$ ZError = '% 1'

#### 개체 오류 코드 - 5600에서 5799

오류 코드	설명
5601	클래스 컨텍스트가 없습니다 : % 1
5602	수퍼 '% 1'을 확인할 수 없습니다
5603	인스턴스 변수 '% 1'이 없습니다
5604	인스턴스 변수 '% 1'은 배열을 지원하지 않습니다
5605	인스턴스 변수 '% 1'에 대한 클래스 컨텍스트가 잘못되었습니다.
5606	잘못된 슈퍼 % 1 을 사용하고 있습니다
5607	참조 변수 '% 1'은 없습니다
5608	참조 변수 '% 1'은 배열을 지원하지 않습니다
5610	참고 매크로가 정의되어 있지 않습니다 : '% 1'
5611	함수 매크로 인수를 찾을 수 없습니다 : '% 1'
5612	참조 매크로, 오른쪽 괄호가없는 '% 1'
5613	매크로의 인수가 너무 많습니다 '% 1'
5614	매크로에 충분한 인수가 없음 : '% 1'
5615	'% 1'속에 종단 % 1 자이 없습니다
5616	# # keyword 뒤에 시작 괄호가 없습니다

5617	잘못된 전처 리기 # # keyword 입니다 : # # % 1
5618	# # % 1 뒤에 종료 괄호가 없습니다
5619	인수 오류 '% 1'
5620	# # % 1 의 Table.Field 입니다
5621	# # % 1 테이블 '% 1'이 없습니다
5622	# # % 1 테이블 '% 1'필드 '% 1'이 없습니다
5623	# # % 1 에 잘못된 인수 '% 1'
5624	# # % 1 (% 1 % 1) 이전 # # % 1 (NEW % 1)가 없습니다
5625	# define 매크로 이름이 없습니다
5626	arglist 에 괄호가 없습니다
5627	# def1arg 1 개 이상의 매크로 매개 변수가 있습니다
5628	매크로 인수가 %로 시작되지 않습니다
5629	인수에 잘못된 문자가 있습니다
5630	마지막 줄에 # #이 계속되고 있습니다
5631	'% 1'이 무시되었습니다; # if 또는 # ifdef 는 처리되지 않습니다
5632	'% 1'에 인수가 없습니다
5633	# if 또는 # elif 인수가 잘못되었습니다
5634	# % 1 에 매크로 이름이 없습니다
5635	include '% 1'이 없습니다
5636	라이브러리 파일 '% 1'이 없습니다
5637	라이브러리 파일 % 1 버전 # % 1 이 없습니다
5638	# sqlcompile 모드가 잘못되었습니다
5639	# # function 오류와 함께 실패했습니다 : \$ ze = % 1

5640	# routine 이 매크로 소스 파일에 이미 지정되어 있습니다
5641	# routine, sql 문 뒤에 지정할 수 없습니다
5642	# routine 잘못된 루틴 명의가 지정되어 있습니다
5643	# # rtnref 을 중첩 호출할 수 없습니다
5644	# # rtnref 에 잘못된 참조가 포함되어 있습니다
5645	이름이 같은 다른 요소가 이미 존재합니다.
5646	# # expression 이 오류와 함께 실패했습니다 : \$ ze = % 1
5647	# define 또는 # def1arg 매크로 이름이 잘못되었습니다 : % 1
5648	# # function 의 사용은 포함된 SQL 로 제한됩니다
5649	이 라인에서 참조하는 (% 1) 매크로가 너무 많습니다. 매크로 정의에서 재귀 호출이 설정되어있을 수 있습니다.
5651	속성을 실행할 수 없습니다
5652	메소드를 설정할 수 없습니다
5653	컴파일된 클래스 '% 1'은 없습니다
5654	메서드 '% 1'은 없습니다
5655	매개 변수 '% 1'은 없습니다
5656	속성 '% 1'은 없습니다
5657	메서드 '% 1'은 반환되지 않습니다
5658	개체 인스턴스가 필요합니다
5659	속성 '% 1'이 필요합니다
5660	검색어 '% 1'이 없습니다
5661	컬렉션 '% 1'은 필수 위해 적어도 하나의 멤버가 있어야합니다
5662	관계의 자식 / 다 측의 속성 '% 1'은 필수 속성이므로, 적어도 하나의 멤버를 가질 필요가

	있습니다.
5701	원하는 이름을 찾을 수 없습니다
5702	왼쪽 괄호를 찾을 수 없습니다
5703	오른쪽 괄호가 없습니다
5704	식의 왼쪽에 오는 등호 기호가 없습니다
5705	불균형 인용입니다
5706	불균형 괄호입니다
5707	불균형 # beginlit .. # endlit 입니다
5710	예상치 못한 # else 합니다
5711	예기치 않은 # elseif입니다
5712	예상치 못한 # endif입니다
5720	예기치 않은 행의 마지막입니다
5721	예기치 않은 파일의 마지막입니다
5730	잘못된 구분 기호입니다
5731	% 1 외부 패키지는 지원되지 않습니다
5732	매크로 중첩 제한을 초과합니다. 순환 매크로 참조를 확인하십시오
5733	변수 % 1 에 대한 이전의 new 가 없습니다
5734	포함된 파일 '% 1'을 찾을 수 없습니다
5740	컴파일하고 있습니다
5741	컴파일이 완료되었습니다!
5742	INT 코드 작성 실패
5743	MAC 코드 작성 실패
5744	모듈의 최대 PCODE 크기를 초과합니다

5745	컴파일이 실패했습니다!
5746	코드 블록을 분할할 수 없습니다. 루틴 '% 1'의 P 코드 % 2 보다 커지고 있습니다.
5747	루틴 '% 1'코드를 분할할 수 없습니다. INT 코드는 없습니다.
5748	# classcontext 문장에 대해 현재 클래스 컨텍스트가 없습니다
5750	개체 '% 1'개방형 보안 위반
5751	메서드 '% 1'에 액세스할 수 없습니다
5752	클래스 '% 1'은 추상 클래스입니다
5753	추상 클래스 '% 1'를 인스턴스화할 수 없습니다.
5754	데이터 타입 클래스 '% 1'를 인스턴스화할 수 없습니다.
5755	개체 '% 1'은 등록되어 있지 않습니다
5756	프로시저 이름 : '% 1'은 잘못된
5757	프로시저 : '% 1'을 찾을 수 없습니다
5758	메서드가 구현되어 있지 않습니다 : % 1
5759	속성은 읽기 전용입니다
5760	개체 인스턴스의 생성에 실패했습니다 : % 1
5761	새 개체 인스턴스를 생성하는 데 실패했습니다 : % 1
5762	클래스 '% 1'은 읽기 전용입니다
5763	'% 1'포함 개체 생성에 실패했습니다
5764	% DeleteExtent 은 '% 1'의 인스턴스 전체를 삭제할 수 없습니다
5765	다른 로케일 시스템 수출을하고 있습니다 : '% 1'
5766	테이블 이름이 잘못되었습니다 : '% 1'
5767	테이블이 이미 존재합니다 : '% 1'
5768	클래스는 이미 존재합니다 : '% 1'

5769	연결 오류 : '% 1'
5770	'% 2'의 '% 1'키 값이 없기 때문에 개체를 여는 데 실패했습니다
5771	'% 2'의 '% 1'키 값이 없기 때문에 개체 삭제에 실패했습니다
5772	컬렉션은 읽기 전용입니다
5773	IDENTITY_INSERT 옵션이 켜진 상태가 아닌 한, ID 속성을 설정할 수 없습니다 : % 1
5774	이전에 할당된 계수기 속성 값을 업데이트할 수 없습니다 : % 1 : % 2
5795	'% 2'에 대한 외래 키 '% 1'참조 객체에 대한 잠금을 가져올 수 없습니다
5796	참조 키 '% 1'참조 객체에 대한 잠금을 가져올 수 없습니다
5797	'% 1'인스턴스 '% 2'key value = '% 3'를 찾을 수 없습니다
5798	단독 액세스를 위해 에쿠스텐토록쿠 실패했습니다 : '% 1'
5799	공유 액세스를 위한 에쿠스텐토록쿠 실패했습니다 : '% 1'

#### 개체 오류 코드 - 5800에서 5999

오류 코드	설명
5800	업데이트시 동시 검증에 실패했습니다 : 개체 버전이 '% 1'과 동일하지 않습니다.
5801	시리얼을 설정할 수 없습니다
5802	속성 '% 1'의 데이터 형식 유효성 검사가 실패했습니다. 값은 "% 2"입니다.
5803	단독 잠금을 획득 할 수 없습니다
5804	읽기 잠금을 획득 할 수 없습니다
5805	Extent '% 1': '% 2'에 고유하지 않은 ID 키가 존재합니다. ID 카운터 위치 = '% 3'
5806	잠금 유형 '% 1'이 올바르지 않습니다
5807	Oref '% 1'이 올바르지 않습니다
5808	키가 고유 없습니다 : % 1

5809	로드하는 객체를 찾을 수 없습니다
5810	삭제할 객체를 찾을 수 없습니다
5811	로드하는 것이 없습니다
5812	NULL 의 ID 입니다
5813	NULL 의 OID 입니다
5814	이미 할당된 Oid 입니다
5815	호출이 너무 많아서 닫을 수 없습니다
5816	트랜잭션 롤백에 실패했습니다
5817	쿼리 속성이 선택되어 있지 않습니다 : % 1
5818	쿼리가 클로즈 않습니다
5819	인수가 너무 많습니다
5820	컬렉션 키 '% 1'이 올바르지 않습니다
5821	쿼리를 인스턴스화할 수 없습니다 : '% 1'
5822	형식 인수 오류 : '% 1'
5823	'% 1'에 의해 참조되고 있기 때문에 객체가 삭제되지 않습니다.
5824	'% 1'에 의해 참조되는 객체가 존재하지 않습니다
5825	% 1 의 인스턴스가 없습니다
5826	클래스 '% 1', '% 2' 인터페이스를 지원하지 않습니다
5827	저장시 잘못된 순회 존입니다
5828	Concurrency 는 0 에서 4 까지의 정수입니다
5829	외래 키 제약 조건 (% 1)가 범위를 볼 때 % 2 참조 무결성 검사에 실패했습니다
5830	외래 키 제약 조건 (% 1)가 객체의 % 2 의 % 3 실패했습니다 (% 4 조회 작업)
5831	외래 키 제약 조건 (% 1)가 객체의 % 2 의 % 3 실패 : 키 % 4 를 참조하는 객체가 최소 1

	개 있습니다
5832	클래스 % 1 ID 값을 요소 중 적어도 하나가 null입니다 : '% 2'
5833	값이 속성 클래스 형식의 인스턴스가 아닙니다 : '% 1 : % 2'
5834	ID 카운터 값이 잘못되었습니다. 콘솔 로그를 확인하십시오 : '% 1'
5835	이미 연결되는 컬렉션은 끊을 수 없습니다
5836	클래스 '% 3'속성 유형이 추상합니다 : '% 1 : % 2'
5837	Null GUID : '% 1'
5838	이 응용 프로그램을 실행하려면 % Development : use 권한이 있어야합니다
5839	CSP 항목 '% 1'을 프로젝트에 추가할 수 없습니다. 동명 대 소문자 다른 항목이 이미 포함되어 있습니다.
5840	파일 '% 1'을 가져올 수 없습니다. 지원되지 않는 형식입니다.
5841	파일 '% 3'행 '% 2'의 오프셋 % 1 이동할 수 없습니다. 줄이 너무 짧습니다.
5842	파일 '% 2'의 줄 번호 % 1 이동할 수 없습니다. 파일이 너무 짧습니다.
5843	사용자 정의 문서 '% 1'를 인스턴스화할 수 없습니다.
5844	사용자 정의 문서 '% 1'은 지원되지 않습니다. 이 네임 스페이스에 사용자 정의 문서 클래스가 없습니다.
5845	항목 '% 1'은 편집 가능한 % 2 이 아닙니다
5846	스튜디오를 사용하려면 % Developer : Use 권한이 있어야합니다.
5847	기본 프로젝트 '% 1'은 가져올 수 없습니다.
5848	기본 프로젝트 '% 1'내보낼 수 없습니다. 이름을 변경하고 내보내십시오.
5849	루틴 '% 1'언어 유형 '% 2'입니다. 지정된 언어와 다릅니다.
5850	이 '% 1'을 프로젝트에 추가 / 프로젝트에서 제거할 수 없습니다. 프로젝트에 이미 패키지 '% 2'가 존재합니다.
5851	라이브러리 클래스를 변경할 수 없습니다

5852	라이브러리 클래스를 저장할 수 없습니다.
5853	잘못된 요소 타입입니다
5854	잘못된 글로벌 참조입니다
5855	잘못된 접두사 oid 입니다
5856	SQLBinding 이 없습니다
5857	스토리지 sql 맵 데이터 이름이 필요합니다
5858	스토리지 sql 맵 이름이 필요합니다
5859	스토리지 sql 지도 행 IDSpec 이름이 필요합니다
5860	스토리지 sql 지도 첨자 이름이 필요합니다
5861	패키지 루틴 접두사가 너무 깁니다.
5862	패키지 글로벌 접두사가 너무 깁니다.
5863	'% 1' 다른 사용자가 편집 중입니다
5864	'% 1' 프로세스 '% 3' 사용자 '% 2' 편집 중입니다
5865	항목 '% 1'은 소스 컨트롤 % 2에서 체크 아웃되지 않습니다
5876	프로젝트 이름이 없습니다
5877	프로젝트 항목 '% 1'의 유형이 잘못되었습니다
5878	프로젝트 항목 이름이 비어 있습니다
5879	가져오기 스트림 데이터가 없습니다
5880	소스 컨트롤 클래스를 생성할 수 없습니다 : % 1
5881	프로젝트 '% 1'은 존재하지 않습니다
5882	새 루틴 이름 '% 1'로 만들 수 없습니다
5883	항목 '% 1'은 쓰기 권한이 없는 데이터베이스에서 매핑된 있습니다
5885	CSP / CSR 페이지 '% 1'의 소스 파일은 읽기 전용으로 표시되어 있기 때문에 읽기

	전용으로 열립니다
5886	컴파일된 사전 클래스를 저장할 수 없습니다.
5887	컴파일된 사전 클래스를 제거할 수 없습니다
5888	새로 컴파일된 사전 클래스를 생성할 수 없습니다.
5889	소스 컨트롤 시스템에 로그인하고 있지 않기 때문에이 작업을 수행할 수 없습니다
5890	루틴 명의 '% 1'이 너무 깁니다
5891	이 프로젝트에 새 이름으로 복사할 수 없습니다.
5892	현재 루틴과 다른 종류의 루틴 '% 1'이 존재합니다. 루틴 명의를 변경하거나, 기존 루틴을 제거하십시오.
5893	파일 '% 1'이 잘못되었습니다. % RO 파일이 중도에 종료하고 있습니다. 루틴 '% 2'가 손상되었을 수 있습니다.
5894	파일의 항목이 너무 많기 때문에 정확하게 항목 목록을 반환할 수 없습니다. 항목 목록은 잘립니다.
5895	항목 '% 1' 다른 네임 스페이스에서 매핑되어 있기 때문에 여기에 저장되지 않습니다.
5896	잘못된 템플릿 모드 '% 1' TEMPLATE, ADDIN, NEW 중 하나입니다.
5897	스튜디오에서 소스 컨트롤 클래스를 변경할 수 없습니다. '% 1'로 잠겨 있습니다.
5898	이 전역의 형식은 너무 길어서 디코딩 수 없습니다.
5899	글로벌 형식이 잘못된 '% 1'을 위해 디코딩 수 없습니다.
5900	패키지 이름 '% 1'이 지정되었지만 실제 패키지 이름이 '% 2'입니다. 대소문자가 일치합니다.
5901	규칙 패밀리 '% 1'이 존재하지 않습니다
5902	규칙 % 1 이 존재하지 않습니다.
5903	규칙 이름이 필요합니다
5904	행 % 3 태그 '<% 1 "'는 속성 '% 2'가 필요합니다.

5905	행 % 3 의 속성 % 1 的 값이 % 2 가 잘못되었습니다.
5906	세션 ID 를 찾을 수 없습니다
5907	세션 ID % 1 이 존재하지 않습니다.
5908	클래스 % 1 : % 2 을 만드는 데 실패했습니다.
5909	행 % 2 태그 <% 1 "에 끝 태그가 없습니다.
5911	문자 % 1 이 설치되어 있지 않기 때문에, 문자 집합 변환을 수행할 수 없습니다.
5912	페이지 '% 1'이 존재하지 않습니다.
5913	HTTP 응답에 잘못된 Content - Type '% 1'이 있습니다
5914	CSP 응용 프로그램 % 1 이 존재하지 않습니다.
5915	라이센스를 할당할 수 없습니다
5916	잘못된 CSP 요청입니다
5917	CSP 는 HTTP 메소드 % 1 을 지원하지 않습니다.
5918	로그아웃되므로이 작업을 수행할 수 없습니다
5919	요청한 동작이 올바르지 않습니다
5920	이 CSP 페이지는 네임 스페이스 '% 1'에서 열기 / 실행해야합니다.
5921	CSP 응용 프로그램 % 1 수행할 대상 네임 스페이스를 지정해야합니다.
5922	응답 대기 시간 초과입니다
5923	% 1 회 리디렉션되었습니다. 리디렉션 루프 것입니다
5924	오류가 발생했습니다 지정 오류 페이지를 표시할 수 없습니다 - Web 마스터에게 통지합니다.
5925	줄 번호 % 1 <SCRIPT LANGUAGE=CACHE> 태그 RUNAT 또는 METHOD 속성을 지정하지 않습니다
5926	HTTP 헤더가 들어 플래시 되었기 때문에, 리디렉션할 수 없습니다

5927	페이지 % 1 클래스 이름에 이미 로드되어 있는 클래스 % 2 가 충돌하여 이 페이지를 로드할 수 없습니다.
5928	행 % 2 태그 <% 1 "의 해석 중에 오류가 발생했습니다
5929	줄 번호 % 1 CSP 지시 문의 구문 분석하는 동안 구문 오류가 발생했습니다
5930	경로 유형 INCLUDE 줄 번호 % 1 파일 이름 지정과 일치하지 않습니다
5931	페이지 이전 OnPreHTTP () 메서드에서 이 메서드를 호출 / 값을 설정할 수 있습니다
5932	Web 서버에서 이 CSP 게토웨이바존에서 동작이 올바르지 않습니다
5933	CSP 서버에 내부 오류가 있습니다 : % 1
5934	행 % 3 위 CSP : OBJECT 태그 '% 2'에서 참조하는 클래스 '% 1'이 정의되어 있지 않습니다.
5935	행 % 2 의 HTML 양식 '% 1'의 이름이 25 자를 초과할 수 없습니다.
5936	행 % 2 의 HTML 양식 '% 1'은 올바른 csp 개체 이름으로 연결하지 않습니다.
5937	행 % 3 의 양식 % 2 에 연결되는 개체 변수 '% 1'이 정의되어 있지 않습니다.
5938	행 % 3 의 양식 '% 2'에서 태그 이름 '% 1'이 유일하지 않습니다.
5939	QUERY 의 SELECT 에 대한 CSPBIND 속성은 행 % 1 영속 객체 참조해야합니다.
5940	행 % 2 의 CSP : OBJECT NAME 속성은 태그 '% 1'에 대한 올바른 식별자가 아니면 안됩니다.
5941	여러 CHECKBOX 태그는 행 % 1 단일 값 필드에 연결할 수 없습니다.
5942	행 % 2 태그 % 1 CSPBIND 특성을 갖지만 결합한 양식은 없습니다.
5943	행 % 1 의 SCRIPT LANGUAGE = SQL 태그는 NAME 과 CURSOR 특성을 모두 가질 수 없습니다.
5944	% 1 속성은 행 % 3 % 2 에 대한 유효한 식별자 여야합니다.
5945	행 % 1 의 MODE 속성은 DISPLAY, LOGICAL, ODBC, SYSTEM 중 하나 여야합니다.
5946	행 % 2 에 SQL CURSOR '% 1'이 중복 정의가 있습니다.

5947	행 % 2 에서 정의되지 않은 SQL 커서 '% 1'을 사용하고 있습니다.
5948	행 % 2 에서 오브젝트 '% 1'중복 정의가 있습니다.
5949	규칙 '% 1'이 중복 정의되어 있습니다.
5950	규칙 '% 2'% 3 번째 클래스 '% 1'은 존재하지 않습니다.
5951	행 % 2 에 csp : search 태그 '% 1'ONSELECT 는 OPTION = POPUP 가 필요합니다.
5952	CSP 규칙 버전이 변경되었습니다. 사용자가 규칙을 다시로드해야합니다.
5953	쿼리 메소드가 값을 반환하지 못했습니다 : % 1
5954	CSP 페이지 잠금에 실패했습니다.
5955	CSPAppList 검색어 : Fetch ()에 잘못된 데이터가 있습니다.
5956	CSP 응용 프로그램 % 2 % 1 디렉터리가 존재하지 않습니다.
5957	CSPPageLookup : 검색 오류입니다.
5958	CSPPageLookup : CLASSNAME 찾을 수 없습니다.
5959	CSPPageLookup : WHERE 을 찾을 수 없습니다
5960	CSPPageLookup : 결과 집합을 생성할 수 없습니다.
5961	문자 % 1 을 변환할 수 없습니다.
5962	새 세션을 할당할 수 없습니다.
5963	잘못된 SysLog 수준 : % 1
5964	줄 번호 % 1 Cache 언어 페이지 지시하는 문장에 의해 변경되었습니다
5965	행 % 2 로 지정되는 Cache 언어 '% 1'이 잘못되었습니다
5966	행 % 2 미지의 문자 집합 '% 1'이 지정되어 있습니다
5967	CSP 하이빠이벤토 요청은 필수 매개 변수를 포함하지 않기 때문에 처리되지 않습니다.
5968	규칙 % 2 CSR : RULE LANGUAGE 속성 값을 % 1 이 잘못되었습니다.
5969	행 % 2 스크립트 태그 언어 '% 1'페이지의 언어와 일치하지 않습니다

5970	줄 번호 % 1 의 정적 SQL 태그 Basic 페이지에서 지원되지 않습니다
5971	1 개의 CSP 라이센스를 1 개의 지명 사용자 라이센스 '% 1'로 교체하는 동안 오류가 발생했습니다.
5972	양식 % 1 에 대한 SaveCallback 잘못된 형식입니다
5973	CSP 페이지 '% 1'이 너무 커서 로드할 수 없습니다. 지원되는 크기는 1.5Mb 까지입니다.
5974	서버 프로세스가 존재하지 않기 때문에 영구 프로세스는 현재 사용할 수 없습니다
5975	다른 프로세스가 잠금을 가지고 있기 때문에, 세션 객체를 잠글 수 없습니다
5976	줄 번호 % 1 Direction 속성이 'forward' 또는 'backward'로하지 않습니다.
5977	csp : search 의 WHERE SELECT, 또는 ORDER 속성 방향 부분은 ASC 또는 DESC 됩니다.
5978	cspSaveMsgEscape 속성 값을 줄 번호 % 1 None, HTML, JS 필요가 있습니다.
5979	세션 ID 가 잘못되었습니다.
5980	Preserve = 1 모드는 실제 Web 서버에서만 지원됩니다.
5981	csp : include 태그는 통합 문서를 지정하기위한 PAGE 특성이 있어야합니다.
5982	행 % 1 SCRIPT LANGUAGE = SQL 태그는 SELECT SQL 명령에만 사용할 수 있습니다.
5983	페이지를 찾을 수 없습니다.
5984	이 응용 프로그램에서 페이지를 실행하려면 인증된 사용 자일 필요가 있습니다.
5985	CSP 세션 서비스 '% 2'에 사용하려고했지만, 세션 서비스 '% 1'에서 시작됩니다.
5986	현재 사용자는 서비스 '% 1'을 수행할 권한이 없습니다.
5987	CSP 페이지에 정의된 메서드는 행 % 2 클래스 메서드이어야합니다.
5988	이 세션에서는 세션 관리 cookie 에만 사용하지만 브라우저에서 CSPCHD 인수를 전달했습니다.
5989	시스템 규칙 (%로 시작하는 이름)와 네임 스페이스 로컬 규칙을 모두 동일한 파일에 정의되지 않습니다.
5990	세션 ID '% 1'을 찾을 수 없습니다.

5991	SOAP 메서드 % 1 을 만들 수 없습니다
5992	SecurityContext 속성 변경은 허용되지 않습니다
5993	CSP 오류 잡기를 사용할 수있는 오류 정보없이 호출됩니다
5994	CSP 프로그램 '% 1'이 지정하는 네임 스페이스 '% 2'가 존재하지 않습니다.
5995	행 % 2 예기치 않은 속성 % 1 입니다

#### 개체 오류 코드 - 6000 6199

오류 코드	설명
6001	파일 '% 1'을 복원할 수 없습니다. OBJ 루틴을 포함합니다.
6002	파일 '% 1%' RO 출력 파일이 없습니다
6003	클래스 형식을 변환할 수 없습니다
6004	XML로 클래스를 내보낼 수 없습니다
6005	XML에서 클래스를 가져올 수 없습니다. 자세한 내용은 '% 1'에 계속됩니다
6006	XML 파일은 인식할 수 있는 가져오기 형식을 포함하고 있지 않습니다
6007	'Content - Length'헤더는 읽기 전용입니다. 설정할 수 없습니다.
6008	'Connection'헤더를 설정할 수 없습니다.
6009	메서드가 지원되지 않습니다.
6010	이미 연결되었습니다.
6011	연결해야합니다.
6012	POP 서버에서 응답이 없습니다 : % 1
6013	메일 서버에 TCP / IP 연결 수 없습니다. 초기 연결이 끊어진 것일 수 있습니다.
6014	TCP / IP 세션은 이미 종료합니다.
6015	POP3 서버가 오류를 보고했습니다 : % 1

6016	% 1 명령에 대한 잘못된 응답입니다 : % 2
6017	메일 읽기 행은 공백이 안됩니다
6018	TCP / IP 세션에 예기치 않은 오류가 발생하였습니다 : % 1
6019	저장 위치를 찾을 수 없습니다
6020	POP 핸들러가 실패했습니다
6021	PUSH 핸들러가 실패했습니다
6022	게이트웨이 실패했습니다 : % 1
6023	쿼리가 없습니다.
6024	잘못된 % qacn 입니다.
6025	게이트웨이 : 잘못된 연결 핸들입니다
6026	게이트웨이 : 문장을 할당할 수 없습니다
6027	NamespaceList 검색어 : Fetch ()에 잘못된 데이터가 있습니다.
6028	매크로 프로세서 오류 : % 1
6029	응답 대기 시간 초과입니다.
6030	SMTP 는 '% 1'속성을 지정해야합니다.
6031	TCP / IP 연결을 열 수 없습니다.
6032	예기치 않은 초기 메시지입니다. 비 SMTP 서버의 가능성이 있습니다 : % 1
6033	SMTP % 1 에 대한 응답 오류 : % 2
6034	% 1 명령을 실행하는 동안 SMTP 서버 연결이 실패했습니다 : % 2
6035	Unicode 시스템에서 출력 문자 집합을 반드시 지정해야합니다.
6036	255 개 이상의 문자는 따옴표로 둘러싸인 메시지가 잘못되었습니다
6037	가져올 것이 없습니다
6038	인스턴스 화하는 데 실패했습니다

6039	RetType 은 VOID 또는 HRESULT 아닙니다
6040	RetType 는 NULL 이 아닌 것입니다
6041	컴파일하는 클래스가 없습니다 : % 1
6042	일과 % 1 개체 코드를 찾을 수 없습니다
6043	데이터베이스에 클래스 정의를 포함하고 있습니다 : % 1
6044	데이터베이스를 탑재할 수 없습니다 : % 1
6045	잘못된 내보낼 디렉토리 이름입니다
6046	데이터베이스가 존재하지 않습니다 : % 1
6047	잘못된 식별자 형식입니다
6048	잘못된 문장 형식입니다 : % 1
6049	잘못된 동적 쿼리 임시 변수 % 1 입니다
6050	매개 변수 값이 잘못된 번호입니다
6051	INTO 절의 생성 오류 :
6052	잘못된 변환 방향 값
6053	직렬화된 데이터가 손상되었습니다.
6054	올바른 % MessageDictionary 가 '% 1'로 지정되어 있지 않습니다
6055	언어가 지정되어 있지 않습니다
6056	출력된 charset 변환 테이블을 찾을 수 없습니다 : % 1
6057	POP3 오류 : % 1
6058	MessageNumber 을 반드시 지정해야합니다.
6059	서버 % 1 TCP / IP 소켓을 열 수 없습니다
6060	사용자가 모니터를 사용하고 있습니다.
6061	모니터가 시작되지 않습니다

6062	모니터가 이미 실행하고 있습니다
6063	모니터에 메모리 할당에 실패했습니다
6064	모니터에서 통계 정보 수집을 사용할 수 없습니다
6065	컬렉션 '% 1'을 열 수 없습니다
6066	'% 1'컴파일 확장 유형이 잘못되었습니다
6067	클래스 인덱스를 다시 작성하는 데 문제가 있습니다
6068	기본 XML 카탈로그 파일 '% 1'을 찾을 수 없습니다
6069	전역 파일 '% 1'% 2 로드 중 오류
6070	특정 E 메일 주소에 SMTP 전송이 실패했습니다
6071	필요한 인수를 찾을 수 없습니다
6072	유효하지 않은 라이센스 키 데이터입니다
6073	라이센스 키 '% 1'을 쓰기 모드로 열 수 없습니다.
6074	ContentTransferEncoding 값이 올바르지 않습니다 : % 1
6075	% 1 블록 번호가 없습니다.
6076	블록 % 1 비트맵 블록은 없습니다.
6077	루틴 '% 1'과 '% 2'를 비교할 수 없습니다. 종류가 다릅니다.
6078	문서 % 2 소스 컨트롤 클래스 액션 % 1 구현이 없습니다
6079	클래스 '% 1'은 합리적인 스튜디오 확장 클래스는 없습니다.
6080	항목 '% 2'% RO 형식 '% 1'타입은 내보낼 수 없습니다.
6081	XML로 내보낸 추상 데이터 문서가 CDATA로 포맷되어 있지 않습니다.
6082	라이센스앗뿌구레도에라 : '% 1'
6083	이 작업에 대한 라이센스가 부여되지 않습니다.
6084	알 수없는 오류가 발생했지만 오류 코드가보고되고 있지 않습니다

6085	SSL / TLS 구성 소켓에 쓸 수 없습니다 : % 1
6086	Content - Type message/rfc822 경우 유일한 부분은 % Net.MailMessage 필요가 있습니다
6087	첨부 E 메일의 Content - Transfer - Encoding 은 '7 bit '또는 '8 bit'이어야합니다.
6088	프록시 '% 1'에서 CONNECT 명령 '% 2'에서 잘못된 응답입니다.
6089	프록시에 대한 CONNECT 명령이 응답 '% 2'에 실패했습니다.
6090	멀티 파트의 Content - Type 경계 특성이 지정되어 있지 않습니다.
6091	예기치 않은 경계선이 MIME 본문 부분에 찾았습니다.
6092	MIME 헤더 형식이 잘못되었습니다.
6093	예상치 못한 메시지 끝에 발견했다. MIME 형식이 잘못되었습니다.
6094	OpenFile 또는 OpenStream 를 사용하여 MIME 메시지 원본을 정의해야합니다.
6095	HTTP 헤더 이름이 너무 길어서 포함할 수 없습니다 '% 1'.
6096	전역 이름 '% 1'이 잘못되었습니다.
6099	UseSTARTTLS 이 참일 경우, SSLConfiguration 를 지정해야합니다.
6100	STARTTLS 는 SMTP 는 지원되지 않습니다 : % 1.
6101	Com 예외 : '% 1'
6102	Com CoClass 기본 인터페이스 정의이 없습니다
6103	Com CoClass 기본 인터페이스가 자동화를 지원하지 않습니다

#### 개체 오류 코드 - 6200에서 6399

오류 코드	설명
6201	객체를 생성할 수 없습니다 : % 1
6202	메시지 처리기를 생성할 수 없습니다 : % 1
6203	예기치 않은 요소입니다

6204	SOAP 메시지에 금지된 처리 지침이 포함되어 있습니다
6205	요소는 적절한 네임 스페이스 여야합니다
6206	버전 오류입니다. 이름 공간은 % 1 이 있어야합니다.
6207	예기치 않은 SOAPACTION 값 : % 1
6208	예기치 않은 특성
6209	특성 번호가 잘못되었습니다.
6210	잘못된 속성 값입니다
6211	특성을 찾을 수 없습니다
6212	잘못된 속성 이름 공간입니다
6213	범위 밖의 특성 네임 스페이스입니다
6214	특성을 식별할 수 없습니다
6215	속성값을 식별할 수 없습니다
6216	지원되지 않는 전송합니다
6217	추가 작업이 실패했습니다
6218	중복 요소입니다
6219	알 수없는 오류
6220	내부 서버 오류
6221	필수 헤더가 지원되지 않습니다 : % 1
6222	웨부메솟도 '% 2'SoapBindingStyle 키워드 '% 1'이 잘못되었습니다.
6223	웨부메솟도 '% 2'SoapBodyUse 키워드 '% 1'이 잘못되었습니다.
6224	Web 서비스의 인수는 타입 % 1 할 수 있습니다
6225	클래스 % 1 에서 DTD 를 생성할 수 없습니다.
6226	WebMethod '% 2'인수 '% 1'은 단순한 형식이나 SOAP 사용이 가능합니다.

6227	서버 응용 프로그램 오류
6228	잘못된 SOAP 메시지입니다
6229	XMLPROJECTION 값이 속성 형식과 다릅니다 : % 1
6230	XMLPROJECTION 속성 값이 잘못되었습니다 : % 1
6231	% XML.Adaptor 의 형식이 잘못되었습니다 : % 1
6232	데이터 형식의 유효성 검증이 태그 % 1 의 값이 % 2 실패
6233	태그 % 1 XML 입력 형식이 적합하지 않습니다.
6234	필요한 태그가 존재하지 않습니다 : % 1
6235	태그에 예기치 않은 네임 스페이스입니다 : % 1
6236	태그를 % 1 참조 ID 를 찾을 수 없습니다 : % 2
6237	XML 입력 예기치 않은 태그가 있습니다 : % 1
6238	배열 태그에 대해 키 특성이 지정되어 있지 않습니다 : % 1
6239	XMLPROJECTION = content 의 속성은 1 개입니다
6240	SERVICENAME 은 SERVICENAME 매개 변수를 무시하여 지정해야합니다.
6241	SOAP WebClient LOCATION 매개 변수는 http 또는 https 전송을 지정해야합니다.
6242	SOAP WebService 에 대한 HTTP 요청에서 예기치 않은 상황이 발생했습니다 : % 1
6243	SOAP WebService 에 대한 HTTP 요청에 대한 응답에서 예기치 않은 CONTENT - TYPE 이 발생했습니다 : % 1
6244	Web 서비스의 위치를 지정해야합니다.
6245	클라이언트 WEB 메서드가 %로 시작하는 인수가 있지 않을 수 있습니다 : % 1
6246	SOAP 요청에 응답이 없습니다.
6247	SOAP 응답의 예기치 않은 인코딩입니다.
6248	SOAP 응답은 SOAP 오류입니다 : % 1

6249	XMLENABLED 클래스에 의해 참조되는 클래스는 % XMLAdaptor 의 하위 클래스 여야합니다 : % 1
6250	컬렉션 속성은 참조되는 클래스의 ELEMENTTYPE 파라미터를 요구합니다 : % 1
6251	메시지 요소 '% 1' XML 네임 스페이스 '% 2'를 찾을 수 없습니다
6252	데이터 형식의 유효성 검사는 태그 % 1 값이 없기 때문에 실패했습니다.
6253	태그 % 1 의 데이터 타입 검증에 실패했습니다. 예기치 않은 태그 <% 2 "을 찾았습니다.
6254	XML 입력 % 1 에 필요한 태그의 형식이 % 2 의 자식으로 잘못되었습니다.
6255	DataSet 레코드의 필드 '% 1'% 2 의 XML 형식이 잘못되었습니다
6256	속성 '% 1'SubstitutionGroup 가 마지막 substitutionGroup 과 일치하지 않습니다.
6257	속성 '% 1'XMLCHOICELIST 에는 리터럴 형식 '% 2'를 포함할 수 없습니다.
6258	속성 '% 2'ENCODING 매개 변수 '% 1'이 잘못되었습니다
6259	이 프로퍼티들은 영속 객체가 아닌 경우, 속성 '% 1'XMLPROJECTION 은 ID 는 없습니다.
6260	데이터 형식의 유효성 검증이 요소 % 3 속성 % 1 의 값이 % 2 에 실패했습니다.
6261	예기치 않은 XMLIGNORENULL 클래스 매개 변수 값 : % 1
6262	XMLIO 속성에 잘못된 값이 있습니다 : % 1
6263	XMLREFERENCE 속성에 잘못된 값이 있습니다 : % 1
6264	XMLTYPECONSTRAINT 속성에 잘못된 값이 있습니다 : % 1
6265	XMLREFERENCE 과 XMLTYPECONSTRAINT 는 속성 클래스 참조에만 지정할 수 있습니다 : % 1
6266	XMLTYPECONSTRAINT 는 속성 XMLREFERENCE = ID 로 지정할 수 없습니다 : % 1
6267	XMLSUMMARY 는, 클래스의 쉼표로 구분된 목록에 있어야합니다.
6268	XMLDEFAULTREFERENCE 잘못된 값입니다.
6269	유형이 지정된 데이터 집합에 CLASS 및 QUERY 를 지정해야합니다.

6270	중복 WebMethod 이름은 허용되지 않습니다 : % 1
6272	QUERYNAME 매개 변수와 클래스 이름 (또는 XMLNAME 재정)는 다를 수 있습니다.
6273	% XML.DataSet 는 QueryName 및 DataSetName 속성을 가질 수 없습니다.
6274	Web services 메서드 % 1 은 SoapNameSpace 는 지원되지 않습니다.
6275	현재 문서가 완성될 때까지 새 XML 문서의 출력이나 % XML.Writer 속성을 변경할 수 없습니다.
6276	루트 요소는 자식 요소가 포함되도록 작성해야합니다.
6277	유형 속성 % 1 은 XML 입력 태그 효과적인 유형은 없습니다 : % 2
6278	XML 출력 문자열은 사용할 수 없습니다.
6279	XML 출력 문자열 길이를 문자열의 최대 길이를 초과합니다.
6280	% XML.DataSet 을 직접 실행하여 쿼리 결과를 얻을 수 없습니다.
6281	클래스 % 2 % 1 % 3 의 하위 클래스와 구별해야합니다.
6282	응답 SOAP 본문이 손상되었습니다.
6283	예상치 못한 세션 헤더 세션 Cookie 입니다.
6284	WS - Security 헤더 오류 : % 1
6285	StartDocument 가 호출되지 않는 한 EndDocument 를 호출할 수 없습니다.
6286	루트 요소, 처리 명령 DOCTYPE 루트 요소에없는 경우가 있습니다.
6287	특성은 요소 또는 루트 요소의 직후에만 호출할 수 있습니다.
6288	엘리먼트 '% 1'% XML.Dataset 스키마가 잘못되었습니다 % 2
6289	데이터 집합 스키마가 지정된 유형이있는 % XML.Dataset 과 일치하지 않습니다 : % 1 % 2 % 3 '= % 4
6290	% XML.Dataset 타자가 지정되지 않은 경우 데이터 집합 스키마는 XML 로 입력해야합니다.
6291	데이터 집합 이름 행 이름 및 XML 이름 공간은 % XML.Dataset XML 스키마와

	일치해야합니다.
6292	% XML.Dataset 열 이름이 중복이 허용되지 않습니다 : % 1
6293	문자 집합 '% 2'에 대한 변환 테이블 '% 1'을 로드할 수 없습니다.
6294	스키마의 메시지 부분을 찾을 수 없습니다 : % 1
6295	XML 스키마 마법사 내부 오류 : % 1
6296	클래스 % 1 XML 내보내기 사이클을 찾았습니다.
6297	XMLSTREAMMODE 속성에 잘못된 값이 있습니다 : % 1
6298	XMLSTREAMMODE 는 문자 스트림이 아니기 때문에 속성 % 1 이 허용되지 않습니다.
6299	XMLNAME 는 속성 % 1 의 유효한 XML 이름을 지정하지 않습니다
6300	XMLFORMAT 잘못된 값입니다
6301	SAX XML 구문 분석 오류 : % 1
6302	행 % 1 오프셋 % 2 XML 메시지 형식이 잘못되었습니다.
6303	Content Handler 가 % XML.SAX.ContentHandler 의 서브 클래스가 없습니다
6304	항목 '% 1'을 내보낼 수 없습니다. XML 내보내기는이 유형의 항목을 지원하지 않습니다. 이 항목을 건너 뛕니다.
6305	사용자 정의 문서 유형 '% 2'를 생성할 수 없기 때문에 항목 '% 1'을 내보낼 수 없습니다. 이 항목을 건너 뛕니다.
6306	CSP 페이지 '% 1'에 연결된 응용 프로그램이 존재하지 않습니다. 건너 뛕니다
6307	페이지 '% 2'에 관련된 CSP 파일 '% 1'이 존재하지 않습니다. 생략합니다.
6308	항목 '% 1'을 부정하거나 내보낼 데이터를 가지고 있지 않습니다. 생략합니다.
6309	클래스 '% 1'은 배치 모드이므로 내보낼 수 없습니다. 이 항목을 건너 뛕니다.
6310	URL '% 1'의 형식이 올바르지 않기 때문에 계속할 수 없습니다.
6311	네임 스페이스 '% 1'에 대한 스키마 정의가 존재하지 않습니다.

6312	클래스 '% 1'데호루토네무스뻬스를 찾을 수 없습니다.
6313	스키마모니카타이뿌 '% 2'(스키마 '% 1'에서)이 잘못되었습니다.
6350	SoapMessageName 키워드는 Web services 메서드 % 1 에 대해서만 지정할 수 있습니다
6351	SoapAction 키워드는 Web services 메서드 % 1 에 대해서만 지정할 수 있습니다
6352	HttpRequester 값이 올바르지 않습니다 : % 1
6353	요소 % 1 에 예기치 않은 특성 : % 2
6354	속성이 문자열이 아니라 XMLPROJECTION = content 를 가지는 경우, 다른 모든 속성은 XMLPROJECTION = attribute 를 가질 필요가 있습니다.
6355	SOAP 메시지 본문이 없습니다.
6356	잘못된 노드 유형입니다 : % 1
6357	부모 노드가 직접 설정하지 않는 것도 있습니다.
6358	트리 스캔 오류입니다. 기대되는 요소입니다.
6359	% SOAPWebRequest 에서는 이진 SOAP 프로토콜을 사용할 수 없습니다
6360	이진 SOAP 프로토콜에 대해 예기치 않은 클래스 % 1 을 받았습니다. % 2 가 필요합니다.
6361	클래스는 XML 을 지원하다해야합니다.
6362	클래스 % 3 XML 스키마 % 1 % 2 에 대한 정의가 중복하고 있습니다.
6363	XML 네임 스페이스 % 1 인코딩의 사용에 일관성이 없습니다.
6364	네임 스페이스 % 1 종류 ElementQualified 정의에 일관성이 없습니다.
6365	SOAP 이진 % 1 의 형식이 잘못되었습니다.
6366	예기치 않은 상위 논리 블록입니다 : % 1.
6367	예기치 않은 SOAP 바이너리 버전 번호 : % 1.
6368	SOAP 이진 메시지 클래스 % 1 정의가 중복하고 있습니다.
6369	SOAP 이진 메시지에서 인덱스 % 1 인 알 수없는 클래스를 개체의 인스턴스를 참조하고

	있습니다.
6370	클래스 % 1 SOAPCLASSNAME 지정이 중복되어 있습니다.
6371	클래스 % 1 SOAPCLASSNAME 에 ServiceName 을 지정해야합니다.
6372	예상치 못한 Content - Type 헤더 필드를 가지는 멀티 파트 MIME SOAP 메시지를 수신했습니다 : % 1. SOAP with Attachments 및 MTOM 만 지원되고 있습니다.
6373	SOAPVERSION 매개 변수로 지정하는 SOAP 버전은 지원되지 않습니다 : % 1.
6374	이 Web 클라이언트는 SOAP 버전 % 1 을 지원하지 않습니다.
6375	SOAP 인코딩 스타일 % 1 은 지원되지 않습니다.
6376	필수 헤더가 지원되지 않습니다.
6378	SECURITYIN 매개 변수 값이 잘못되었습니다 : % 1
6379	WS - Security 헤더가 필요합니다.
6380	인증서 파일의 형식이 잘못되었습니다 : % 1.
6381	WS - Security 의 암호화 알고리즘을 지원하지 않습니다 : % 1.
6382	키 암호화를 실패했습니다 : % 1.
6383	암호화에 실패했습니다 : % 1.

#### 개체 오류 코드 - 6400에서 6599

오류 코드	설명
6401	요소 '% 1'속성 '% 2'가 잘못되었습니다
6402	요소 '% 1'속성 '% 2'가 잘못된 값을 % 3 입니다.
6403	요소 '% 1'은 잘못된 속성이 포함됩니다
6404	요소 '% 1'이 잘못되었습니다
6405	요소 '% 1'값 '% 2'가 잘못되었습니다
6406	지정된 네임 스페이스 '% 1'이 잘못되었습니다. '% 2'이어야합니다.

6407	스키마 유형을 짐작할 수 없습니다. 효과적인 대응을 찾을 수 없습니다
6408	메시지 유형을 짐작할 수 없습니다. 효과적인 대응을 찾을 수 없습니다
6409	인코딩 '% 1'은 지원되지 않습니다
6410	요소 '% 1'필수 속성 '% 2'가 없습니다
6411	요소 '% 1'을 찾을 수 없습니다
6412	요소 '% 1'- 작업 % 3 을 위해 % 2 를 결정할 수 없습니다
6413	요소 '% 1'- 찾습니다 % 2 % 3
6414	요소 '% 1'- 이름이 중복 '% 2'
6415	요소 '% 1'- 지원되지 않는 전송 '% 2'
6416	요소 '% 1'- 인식되지 않는 % 2 요소 '% 3'
6417	요소 '% 1'- 메시지 '% 2'유형 또는 요소의 특성이 일부로 지정해야합니다
6418	엘리먼트 '% 1'- 리테라루엔코딩구 대한 메시지 '% 2'매개 변수를 찾을 수 없습니다
6419	요소 '% 1'- 작업 % 3 에 대해 % 2 일치하지 않습니다
6420	요소 '% 1'- 마루찌빠토바인딩구는 지원되지 않습니다
6421	요소 '% 1'- ParameterOrder 매개 변수 개수가 잘못되었습니다
6422	targetNamespace = % 1 에 대한 WSDL 이름 공간을 정의하지 않습니다.
6423	targetNamespace = % 1 의 SOAP 네임 스페이스가 정의되어 있지 않습니다.
6424	엘리먼트 '% 1'- 메시지 '% 2'유형과 요소 속성은 동시에 일부로 지정할 수 없습니다.
6426	클라이언트 클래스와 서비스 클래스를 동일한 패키지에 포함될 수 없습니다 : % 1.
6440	% SOAP.Configuration XData 블록 % 2 에서 예기치 않은 루트 요소 % 1 이 검색되었습니다.
6441	% SOAP.Configuration XData 블록 % 2 에서 예기치 않은 요소 % 1 이 검색되었습니다.
6442	% SOAP.Configuration XData 블록의 이름이 중복됩니다 : % 1

6443	SOAP 클래스 % 2 에 대해 중복 구성 이름 % 1
6444	구성에서 서비스의 SOAP 클래스 이름이 지정되어 있지 않습니다 : % 1
6445	구성에서 메서드 요소에 대한 메서드 이름이 지정되어 있지 않습니다 : % 1
6446	구성 % 2 에서 중복 메서드 이름 % 1
6447	% SOAP.Configuration XData 블록 % 2 에서 WS - Policy 네임 스페이스에서 예기치 않은 요소 % 1 이 검색되었습니다.
6448	구성의 Name 속성 % 1 % SOAP.Configuration XData 블록 % 2 와 일치하지 않습니다.
6449	SOAP 구성 클래스 이름 % 1 이 잘못되었습니다.
6450	% SOAP.Configuration 클래스 % 2 구성을 찾을 수 없습니다. % 1
6451	정책 어설션, % 1 % SOAP.Configuration 클래스 % 2 에서 텍스트의 자식 요소를 가지고 있지 않을지도 모릅니다
6452	구성 % 1 에서 정책 분석하는 동안 내부 오류가 발생했습니다 : % 2
6453	지원하지 않는 아사촌네무스뻬스 "% 1", assertion = % 2 구성 = % 3
6454	구성 % 1 에서 지원되는 대체 정책이 없습니다.
6455	구성 % 2 정책 어설션 % 1 은 지원되지 않습니다.
6456	구성 % 2 정책 어설션 % 1 인식할 수 없습니다.
6457	구성 % 2 정책 어설션 % 1 wsp : Policy 자식 요소를 가질 수 없습니다.
6458	구성 % 3 정책 어설션 % 1 에 지원되지 않는 매개 변수 % 2 이 있습니다.
6459	구성 % 2 정책 어설션 % 1 헤더 매개 변수는 Namespace 속성이 필요합니다.
6460	구성 % 2 정책 어설션 % 1 헤더 매개 변수는 Name 특성이 있어야합니다.
6461	구성 % 3 정책 어설션 % 1 이 중첩된 정책 어설션 % 2 를 지원하지 않습니다.
6462	구성 % 2 정책 어설션 % 1 이 중첩된 정책이 필요합니다.
6463	구성 % 2 정책 어설션 % 1 예상한 이름 공간에 없습니다.

6464	구성 % 2 % 1 하나만을 지정할 수 있습니다.
6465	구성 % 2 토큰 % 1에 대한 주장 매개 변수가 지원되지 않습니다
6466	구성 % 3 assertion % 2 토큰 % 1은 지원되지 않습니다
6467	구성 % 2 % 1에 AlgorithmSuite assertion이 필요합니다.
6468	구성 % 2 토큰 % 1 형식이 잘못되었습니다.
6469	구성 % 2 sp : IncludeToken % 1에 대해 예기치 않은 값입니다.
6470	구성 % 3 % 1은 하나의 % 2 토큰이 필요합니다.
6471	구성 % 3 % 1 최소한 하나의 토큰이 필요합니다.
6472	% SOAP.Configuration XData 블록 % 2의 % 1 요소가 예상한 네임 스페이스에 없습니다
6473	구성 % 3 요소 % 2 중 예기치 않은 속성 % 1이 있습니다.
6474	구성 % 1 sp : X509Token 위해 cfg : FindField 과 cfg : FindValue 중 하나가 지정된 경우, 그들을 모두 설정해야합니다.
6475	구성 % 2 sp : X509Token에 cfg : FindField 예기치 않은 값입니다.
6476	구성 % 1 wsp : PolicyReference 요소에 대한 로컬 URI 속성이 없습니다.
6477	구성 % 2 wsp : PolicyReference 요소에 대한 URI 속성 % 1,이 정책을 참조하지 않습니다.
6478	sp : Username 토큰은 sp : SignedParts 또는 sp : EncryptedParts assertion에서 지원하는 토큰에 대해 잘못되었습니다.
6479	구성 % 2 cfg : wsdlElement = "% 1" wsdlElement 대한 적절한 값이 지정되어 있지 않습니다
6501	인식되지 않는 XSD 유형 '% 1'
6502	지정된 XSD 유형 '% 1'에 해당하는 Cache의 유형을 결정할 수 없습니다

#### 개체 오류 코드 - 6600에서 6799

오류 코드	설명

6601	BeanName 가 필요합니다.
6602	RootDir 가 필요합니다.
6603	ClassPath 가 필요합니다.
6604	App Server Home 이 필요합니다 (APPERVERHOME 가 "").
6605	Java Home 이 필요합니다.
6606	Path 가 필요합니다.
6607	ServerType 이 필요합니다.
6608	QuickStatement 인터페이스만 지원합니다.
6609	% 1 Persistent (영구)에서도 Session Bean 도 아닙니다. ClassList % 2 입니다. ClassList 는 Persistent (영구) 또는 Session Bean 클래스만을 포함해야합니다.
6610	ClassList 가 프로젝션 또는 루틴의 호출로 지정해야합니다. ClassList 이 하늘이 안됩니다.
6611	서버 이름 % 1 은 EJB 마법사 정의되어 있지 않습니다. WEBLOGIC, WEBLOGIC7, WEBLOGIC8, JBOSS, JBOSS3, PRAMATI. 중에서 1 개를 선택하십시오. JBOSS 은 JBoss 2.4.3 및 2.4.4 에 대한 코드를 생성합니다. JBOSS3 는 JBoss 3.X 에 대한 코드를 생성합니다. WebLogic 6.1 WEBLOGIC, WebLogic 7.0 WEBLOGIC7, WEBLOGIC 8.1 에서 WEBLOGIC8 을 사용합니다. 릴리스 노트를 참조하여 지원되는 서버 목록을 확인하십시오.
6612	Class = % 1 CMP 생성기가 실패했습니다. CMP 생성, 기본 키를 가지는 클래스에서만 지원됩니다.
6613	일반적인 CPP 출력이 설정되어 있지 않습니다
6614	getClassMethodsError : % 1 className = % 2
6615	getClassPropertiesError : % 1 className = % 2
6616	getClassQueriesError : % 1 className = % 2
6617	getEJBClassNameError : % 1 className = % 2
6618	getEJBClassNameError : % 1
6619	일반적인 출력이 설정되어 있지 않습니다

6620	공통 언어 생성기 개체가 설정되어 있지 않습니다
6621	EJB Easy 프로젝트는 Windows에서만 지원됩니다. UNIX®에서는 EJB를 사용합니다.
6622	PersistenceType, BMP 또는 CMP 중 하나 여야합니다.
6623	Class = % 1 CMP 생성에 실패했습니다. CMP 생성에 필요한 모든 등록 정보가 CMP와 호환되는 경우에만 수행할 수 있습니다. Property = % 2 CMP와 호환되지 않습니다.
6624	ClassList = % 1의 EJB 생성에 실패했습니다. ClassList는 세션 bean이 아닌 적어도 1개의 영속 클래스들을 포함해야합니다.
6625	WebLogic은 연결 풀에 있는 연결을 확인하기 위해 테스트 테이블을 정의해야합니다
6626	클래스 % 1 투영할 수 없습니다. 투영을 중지하고 있습니다. 클래스의 슈퍼 클래스가 % RegisteredObject 아니라 슈퍼 클래스의 메소드가 모두 서버에만이 클래스에 클래스 메소드가 아닌 메소드가 포함되는 경우, 이 클래스는 투영할 수 없습니다. 클래스의 슈퍼 클래스가 NULL의 경우, 이 클래스에 포함되어 있는 클래스 메서드만 필요가 있습니다.
6627	% 1 TRANSACTIONISOLATION 대한 유효한 값이 아닙니다. 유효한 값은 TRANSACTION_READ_UNCOMMITTED과 TRANSACTION_READ_COMMITTED입니다.
6628	종류 % 1 투영할 수 없습니다. 투영을 중지하고 있습니다. 투영되는 클래스의 메서드는 모두 왼쪽의 슈퍼 클래스와 같은 서명이 있어야합니다. 메서드 % 2와 충돌합니다.
6629	클래스 % 1 투영할 수 없습니다. 투영을 중지하고 있습니다. 투영되는 클래스의 모든 속성은 왼쪽의 슈퍼 클래스와 같은 선언이 있어야합니다. 속성 % 2와 충돌합니다.
6630	클래스 % 1 투영할 수 없습니다. 투영을 중지하고 있습니다. 프로젝션하려면 왼쪽의 슈퍼 클래스 % 2 % Library.RegisteredObject이나 클래스 % 1 "정적"(클래스 메서드만을 가지고 속성 및 인스턴스 메서드를 가지지 않는다)이어야합니다.
6631	영속 클래스 % 1 투영할 수 없습니다. 투영을 중지하고 있습니다. 예상되는 영속 클래스의 왼쪽에 있는 슈퍼 클래스 % 2 % Library.Persistent 필요가 있습니다.
6632	클래스 % 1 마지막 수업은 없습니다. 클래스를 다시 컴파일하고 다시 시도하십시오.
6633	클래스 % 1을 EJB에 투영할 수 없습니다. 투영을 중지하고 있습니다. 투영되는 클래스의 자식 테이블을 모두 사용할 필요가 있습니다. 자식 테이블 % 2가 잘못되었습니다.
6634	getClientClassDefError : % 1 className = % 2

6635	QueryGetInfoEror : % 1, className = % 2, query = % 3
6636	클래스 % 1 데이터 형식 투영 수 없습니다.
6637	/ (백슬래시)를 포함하기 위하여 형식 플래그가 올바르지 않습니다. 잘못된 형식 플래그는 % 1 입니다.
6638	% 1 클래스 % 2 돌려주려고하고있는 목록은 너무 큽니다.
6639	getCountMethodsError : % 1 className = % 2
6640	getCountPropertiesError : % 1 className = % 2
6641	getCountQueriesError : % 1 className = % 2
6642	속성 % 2 이 필요하지만, EJB 마법사에서 아직 지원되지 않으므로 % 1 의 EJB 영사가 중단됩니다.
6643	클래스 % 1 내보낼 수 없습니다. 내보낼 수있게하려면 % Complier.LG.Exportable 를 확장해야합니다.
6645	% 1 : % 2 className = % 3
6646	JAVAPACKAGE 매개 변수가 clientname 매개 변수와 충돌합니다. JAVAPACKAGE 은 % 1 clientname 매개 변수는 % 2 입니다
6647	서버측 코드를 생성할 수 없습니다. cpp_generator 를 사용하여 클라이언트 코드를 생성하십시오
6648	생성에 문제가있는 클래스 % 2 에 의존하기 때문에 클래스 % 1 의 코드는 생성할 수 없습니다.
6649	% 3 의 이유로 발생이 할 수없는 클래스 % 2 에 의존하기 때문에 클래스 % 1 의 코드는 생성할 수 없습니다.
6650	serveronly 클래스인 클래스 % 2 에 의존하기 때문에 클래스 % 1 의 코드는 생성할 수 없습니다.
6651	클래스 % 1 예측 가능하지 않고, serveronly 않은 것에 의존하기 때문에 클래스 % 1 의 생성을 건너 뛕니다. 다음은 자세한 정보입니다. % 2
6653	클래스 % 1 빈 JavaBlock 있습니다

6654	클래스 % 1 여러 JavaBlock 있습니다
6655	슈퍼 % 2 컬렉션이므로, % 1 을 투영할 수 없습니다
6656	메서드 % 2 ByRef 인수를 가지므로 % 1 POJO 로 투영 수 없습니다
6657	% 2 을 반환 유형 또는 인수 유형의 추상 스트림을 가지므로 % 1 을 투영 수 없습니다
6658	맨 왼쪽의 슈퍼 % 2 스트림이기 때문에 % 1 을 투영할 수 없습니다
6659	클래스 % 1 캐시에 대한 읽기 대기 시간이 초과되었습니다.
6701	이미 연결되어 있습니다
6702	PID 값을 찾을 수 없습니다.
6703	잘못된 PID 값
6704	대상 디버거가 이미 종료합니다
6705	대상 휴식을 게시할 수 없습니다
6706	CSP 서버 접속 오류 : % 1
6707	연결되어 있지 않습니다
6708	대상의 연결 오류입니다
6709	대상이 중지되지 않습니다
6710	대상에 연결할 수 없습니다
6711	잘못된 디버깅입니다 : % 1
6712	중단점 '% 1'에 대한 매핑이 없습니다
6713	대상을 시작하는 데 실패했습니다
6714	캐시 디버거 오류 : % 1
6715	잘못된 PID 값 '% 1'

#### 개체 오류 코드 - 6800에서 6999

오류 코드	설명
-------	----

6901	XSLT XML 변환 오류 : % 1
6902	Error Handler 가 % XML.XSLT.ErrorHandler 의 서브 클래스가 없습니다
6903	Output Stream 이 % BinaryStream 의 서브 클래스가 없습니다
6904	Result Handler 가 % XML.XPATH.ResultHandler 의 서브 클래스가 없습니다
6905	Input Stream 이 % BinaryStream 의 서브 클래스가 없습니다
6906	% New ()를 직접 호출해야 합니다. 팩토리 메소드 'Create ...'를 사용하십시오

#### 개체 오류 코드 - 7000에서 7199

오류 코드	설명
7001	TSQL 컴파일러 오류 : % 1
7002	TSQL : % 1
7003	ISQL 컴파일러 오류 : % 1
7004	ISQL : % 1
7005	TSQL 쿠에리비루다에서 "% 1"을 기대하고 있었습니다 (얻은 것은 "% 2")
7006	쿼리 결과를 변수에 저장할 수 없습니다
7011	TSQL 언어 모드는 절차 차단해야합니다 : '% 1 : % 2'
7050	클래스 정의의 개방형 오류가 발생했습니다 "% 1": "% 2"
7051	알 수없는 입력합니다 : "% 1"
7052	Read : "]"이 없습니다
7053	읽기 : 문자열의 마지막에 따옴표가 없습니다
7054	Read : "% 1"근처에 구문 오류가 있습니다
7055	쿼리 실행 : 프로 시저 "% 1"이 쿼리는 없습니다
7056	쿼리 실행 : 인수 "% 1"은 이미 ( "% 2"으로) 사용하고 있습니다
7101	지정된 검색 위치 (% 1)는 파일의 끝 (% 2)보다 뒤에 있습니다.

7102	FileStream Mode % 1 은 읽기 전용 모드의 설정이 포함되지 않습니다
7103	FileStream Mode % 1 에 쓰기 모드 설정이 포함되지 않습니다
7104	이 MetaStream 결합되는 대체 입력 스트림이 없습니다
7105	CharEncoding '% 1'에 매핑하는 변환 테이블이 없습니다
7106	IO 스트림 클래스 % 1 닫히지 않습니다
7107	대행 IO 스트림 클래스 % 1 찾을 수 없습니다
7108	타입 % 1 개체는 스트림 개체가 없습니다
7109	스트림 '% 1'오픈에서 % 2 초 후에 시간 초과되었습니다.
7110	소켓 '% 1'에서 수신 대기 % 2 초의 시간이 초과되었습니다
7150	Telnet 옵션 % 1 설정되지 않았습니다
7151	telnet 핸드 셰이크 오류가 있습니다 (상태 = % 1, 현재의 바이트 수 = % 2)
7152	telnet 초기화 핸드 시도가 시간 초과되었습니다.

#### 개체 오류 코드 - 7200에서 7399

오류 코드	설명
7200	타입 값 '% 1'IsValidDT 검증에 실패했습니다
7201	타입 값 '% 1'의 길이가 % 2에서 허용되는 MAXLEN 보다 길어지고 있습니다
7202	타입 값 '% 1'의 길이가 % 2에서 허용되는 MINLEN 보다 짧습니다
7203	타입 값 '% 1'% 2에서 허용되는 MAXVAL 를 초과합니다
7204	타입 값 '% 1'% 2에서 허용되는 MINVAL 보다 작습니다
7205	타입 값 '% 1'VALUELIST '% 2'에 지정된 값이 없습니다
7206	타입 값 '% 1'은 합리적인 불리언가 없습니다
7207	타입 값 '% 1'은 합리적인 숫자가 아닙니다

7208	타입 값 '% 1'은 합리적인 타임 스탬프 형식은 없습니다
7209	타입 값 '% 1'패턴 '% 2'와 일치하지 않습니다
7300	출력을 위해 % 1 로그 파일을 열 수 없습니다
7301	Backup.General.ExternalFreeze 은 이미 설정되어있는 스위치 10 또는 13에서 실행할 수 없습니다
7302	모든 구성원의 TCP 정보의 위치를 확인할 수 없습니다.
7303	다른 클러스터 구성원 저널 파일을 전환할 수 없습니다
7304	저널 파일을 변경할 수 없습니다. Status = % 1
7305	로컬 저널 파일을 변경할 수 없습니다. Status = % 1
7306	시스템을 종료할 수 없습니다
7307	저널 마커를 배치할 수 없습니다
7308	작업 % 1 이 존재하지 않습니다
7309	백업은 현재 실행 중입니다
7310	작업 % 1 을 열 수 없습니다
7311	작업 % 1 의 백업은 기록되지 않습니다
7312	\$ zversion (1) 알 수 없는 플랫폼입니다
7313	로그 파일의 목록 작성 오류 : % 1
7314	작업 목록의 작성 오류 : % 1
7315	% 1에서 기본 디렉토리를 확인할 수 없습니다
7316	로그 파일을 저장할 디렉토리를 만들 수 없습니다 : % 1
7317	% 1의 기본 디렉토리를 확인할 수 없습니다
7318	백업 출력 디렉터리를 생성할 수 없습니다 : % 1
7319	백업을 위한 데이터베이스 목록을 설정할 수 없습니다.

7320	알 수없는 백업 유형입니다 : % 1
7321	데이터베이스 % 1 은 존재하지 않습니다
7322	CACHETEMP 은 백업에 포함할 수 없습니다
7323	데이터베이스 목록 작성 오류 : % 1
7324	% 1 은 현재 백업 목록에 없습니다
7325	백업 작업을 시작할 수 없습니다.
7326	IJC 장치 초기화에 실패했습니다 : % 1
7327	BACKUP ^ DBACK 실패를 반환했습니다.
7328	임베디드 시스템 태스크는 변경할 수 없습니다
7329	잘못된 백업 유형입니다 : % 1
7330	작업 이름이 % New 에 대한 인수로 지정되어 있지 않습니다
7331	작업 이름은 알파벳 문자 밖에 사용할 수 없습니다
7332	작업은 이미 존재합니다

#### 개체 오류 코드 - 7400에서 7599

오류 코드	설명
7400	TASKMGR 는 이미 실행 중입니다
7401	작업 (% 1)를 열 수 없습니다
7402	선택된 사용자 (% 1)는 사용되지 않습니다
7403	작업 (% 1)의 실행은 예정되어 있지 않습니다
7404	1 일 여러 번하지 않지만 DailyIncrement 은 0 입니다
7405	작업을 만들 수있는 권한이 없습니다
7406	사용자가 존재하지 않습니다 (RunAsUser % 1)
7408	DailyEndTime 는 DailyStartTime 후에해야합니다

7409	EndDate 는 StartDate 후에해야합니다
7410	TimePeriodDay 는 NULL 이거나 유효하지 않은 1 ~ 7 (% 1)의 값을 포함하고 있습니다
7411	출력 디렉터리가 존재하지 않습니다
7412	파일 이름은 사용할 수 없습니다
7413	작업 클래스가 필요하지만, NULL입니다
7414	작업 클래스 (% 1) % 2 에 존재하지 않습니다
7415	삭제 작업을 찾을 수 없습니다
7416	일정에서 알 수없는 문제입니다. New Time = Last Time
7418	작업을 일시 중지으로 표시할 수 없습니다 (SQLCODE = % 1)
7419	작업을 다시 시작 됨으로 표시할 수 없습니다 (SQLCODE = % 1)
7420	잘못된 일시 정지 플래그 (FLAG = % 1)
7421	메일을 보낼 수 없습니다. 메일 서버가 정의되어 있지 않습니다.
7422	메일을 보낼 수 없습니다. 전자 메일 주소가 정의되어 있지 않습니다.
7423	오류 % 2 를 위해, (% 1) 구성을 업데이트할 수 없습니다
7424	전자 우편 (% 1)를 보낼 수 없습니다
7425	작업을 삭제할 수 없습니다. 처음으로 이동하면 모든 실행을 지웁니다
7426	월요일 ~ 일요일에서 실행 날짜를 적어도 1 일 선택합니다
7427	오프셋을 반복 작업은 양수 여야합니다
7428	그 달 (% 1) 잘못된 날짜
7429	잘못된 주별 오프셋 1 ~ 5 를 사용하고 있습니다
7430	수 (DailyFrequencyTime)이 잘못되었습니다. 0 또는 1 을 사용하십시오.
7450	작업 작업 중입니다
7451	작업 작업 함정 해제 오류 (% 1)

7452	작업 작업의 설정 오류 (% 1)
7453	작업 작업 시간 초과 오류
7454	작업 작업 후처리 오류 (% 1)
7460	테이프 장치를 입력할 필요가 있습니다.
7461	유효한 날짜를 입력하세요.
7500	SSH % 3 오류 '% 1': % 2

#### 개체 오류 코드 - 7600에서 7799

오류 코드	설명
7604	글로벌 노드 종류 '% 1'과 충돌합니다
7607	% 1의 원래 데이터 = % 2 새 값을 % 3
7608	새 데이터에 % 1 = % 2 가 있지만 원래 글로벌 이것은 존재하지 않습니다
7700	잘못된 매니 페스트 지정 '% 1'
7701	잘못된 표현 '% 1': % 2
7702	잘못된 특수 변수 '% 1'

#### 개체 오류 코드 - 7800에서 7999

오류 코드	설명
7800	JavaScript 오류 : % 1
7801	JavaScript 엔진 오류 : % 1

#### 기타 오류 코드

오류 코드	설명
DisplayStringLoaderError	DisplayStringLoader 오류 '% 1'
DisplayStringLoaderException	DisplayStringLoader 예외 '% 1'
DomainOrFilesEmpty	도메인 매개 변수 또는 파일 매개 변수가 비어 안됩니다

오류	오류
ErrDisplayStringNotFound	Id = '% 1', domain = '% 2', language = '% 3' DisplayString 가 정의되지 않습니다.
ErrNoSaveMasterStrings	MasterLanguage '% 1'에서 파일 % 2에 XData MasterStrings 저장에 실패했습니다.
MasterStringsNewer	% 1 XData MasterStrings % 2 문자열보다 새로운 문자열입니다 - 첫번째 차이 = % 3
MasterStringsOlder	% 2 의 % 1 문자열 XData MasterStrings 보다 새로운 문자열입니다 - 첫번째 차이 = % 3
MessageDomainNotFound	도메인 매개 변수에 지정된 도메인은 어떤 메시지 파일에서 발견되지 않았습니다
NoStatusCode	(오류에 대한 설명은 없습니다.)
OK	OK
STATUS	STATUS
UnknownStatusCode	미지의 상태 코드 :
경고	경고
XMLImportLocation	(행 % 1 문자 % 2 종료)

## CSP 오류

### CSP 오류 코드, 오류 메시지 및 오류 조건

오류 코드	오류 메시지	오류 조건
5902	규칙 % 1 이 존재하지 않습니다.	% apiCSP 호출 규칙에 속성을 추가할 때 존재하지 않는 규칙 이름을 지정하면 발생합니다.
5903	규칙 이름이 필요합니다.	규칙을 추가하거나 제거할 때 해당 규칙의 이름을 지정하지 않은 경우 발생합니다.
5904	행 % 3 태그 '<% 1 %>'는 속성 '% 2'가 필요합니다.	CSP 페이지의 태그에 필요한 속성을 지정하지 않은 경우에 발생합니다. 이 필수 특성을 지정하지 않으면 페이지를 컴파일할 수 없습니다.
5905	행 % 3 의 속성 % 1 의 값이 % 2 가 잘못되었습니다.	CSP 페이지의 속성 값이 잘못된 경우에 발생합니다. 예를 들면, <b>&lt;script language="Cache" runat="XXXXX"&gt;</b> 를 정의하면, <b>runat</b> 값이 잘못되었습니다. CSP 컴파일러는 이 페이지를 컴파일할 수 없으며, 이 오류를 보고합니다.
5906	세션 ID 를 찾을 수 없습니다.	% New 메서드에서 세션 ID 를 지정하지 않고 % <b>CSP.Session</b> 인스턴스를 만들려고 하는 경우에 발생합니다. 예를 들면, <b>Set session = # # class (% CSP.Session) % New ()</b> 를 지정하면 이 오류가 발생하지만, <b>Set session = # # class (% CSP.Session) % New (1234)</b> 을 지정하면 세션 ID <b>1234</b> 전달되기 때문에 오류가 발생하지 않습니다.
5907	세션 ID % 1 이 존재하지 않습니다.	기존 % <b>CSP.Session</b> 를 로드할 때, Caché 에 저장되지 않은 세션 ID 를 % <b>OpenId</b> 에 전달하려고하면 발생합니다.
5908	클래스 % 1 : % 2 을 만드는 데 실패했습니다.	CSP 페이지에 해당하는 클래스를 생성할 수 없는 경우, CSP 컴파일러에서 이 오류가 보고됩니다.

5909	행 % 2 태그 <% 1 "에 끝 태그가 없습니다.	규칙 정의에 닫는 태그가 요구되는 태그를 연 후, 종료 태그를 작성하지 않은 경우 발생합니다.
5911	문자 % 1 이 설치되어 있지 않기 때문에, 문자 집합 변환을 수행할 수 없습니다.	이 페이지를 출력하기 위해서 CSP 페이지에 지정된 문자 집합이 Caché 에 설치되지 않은 경우에 보고됩니다. 이것은 <b>OnPreHTTP</b> 메서드 % <i>response.CharSet</i> 속성에 지정된 문자 집합 또는 <b>&lt;csp:content charset="xxx"&gt;</b> 태그 또는 <b>&lt;meta http-equiv="Content-Type" content="text/html; charset=xxx"&gt;</b> 태그를 사용하여 원하는 페이지에 지정된 문자 집합입니다. 클래스 % <b>CSP.Page</b> 의 <i>charset</i> 속성을 참조하십시오. 오류보고 문자 집합을 사용할지 여부를 확인하고 사용하려면 해당 문자 집합을 Caché 에 설치하거나 <b>OnPreHTTP</b> 메서드 % <i>response.CharSet</i> 속성을 설정합니다.
5912	CSP 페이지 % 1 이 존재하지 않습니다.	존재하지 않는 CSP 페이지를 요청하면 발생합니다. URL 의 철자가 틀리거나 다른 CSP 페이지에 대한 링크가 잘못되었습니다. 이 페이지가 서버에 있는지 여부를 확인하고 존재하지 않으면 원본을 확인합니다. 페이지가 존재하는 경우는 CSP 응용 프로그램이 올바른 디렉터리를 가리 키도록 설정되어 있는지, CSP 파일이 디스크에 있는지 확인합니다. 이 오류는 <b>autocompile</b> 옵션이 켜진 상태에서, CSP 엔진이 페이지에 액세스하여 컴파일하려고하면 파일이 없는 경우에 발생합니다.
5914	CSP 응용 프로그램 % 1 이 존재하지 않습니다.	URL 응용 프로그램 부분이 CSP 응용 프로그램 목록에서 찾을 수 없는 경우 발생합니다. 예를 들면, / <b>cspx</b> / <b>samples</b> / <b>menu.csp</b> 는 페이지를 로드할 때, <b>csp</b> 대신 <b>cspx</b> 를 입력하면, Caché 는 올바른 CSP 응용 프로그램을 찾을 수 없습니다. 시스템 관리 포털 [ <b>CSP 응용 프로그램페이지</b> ( 홈 > "보안 관리" > [CSP 응용 프로그램] ) 응용 프로그램의 목록을 확인하고

		명령에 오류가 있는지 확인합니다.
5915	라이센스를 할당할 수 없습니다.	라이센스 제한에 도달하고 있기 때문에, CSP 세션에 대한 새로운 요구가 허용되지 않는 경우 발생합니다. CSP 응용 프로그램에서 지정되어 있는 CSP 세션의 기본 제한 시간을 줄이거나 추가 라이센스를 구입할 필요가 있습니다.
5916	잘못된 CSP 요청입니다.	개인 페이지에 접근하기 위해, 이 페이지에 액세스할 수 있는 암호화된 토큰을 가진 CSP 페이지에서 리디렉션 대신 URL을 직접 지정으로 액세스하려고하면이 오류가 발생합니다. 또한 암호화된 토큰이 유효하지 않은 경우에도 이 메시지가 보고됩니다.
5917	CSP는 HTTP 메소드 % 1을 지원하지 않습니다.	지원되지 않는 HTTP 메서드를 사용하려고하면 발생합니다. 지원되는 HTTP 메서드는 GET, POST, HEAD의 3 가지입니다. 현재 다른 HTTP 메서드는 CSP 서버에서 지원되지 않습니다. 또한 CSP 서버와 통신하는 CSP 게이트웨이 버전이 맞지 않는 경우에도 이 메시지를 얻을 수 있습니다.
5918	로그아웃 때문에 작업을 수행할 수 없습니다.	CSP 요청에 암호화된 데이터가 들어있어도 세션이 새로이므로 해독 키가 암호화 데이터와 일치하지 않는 경우에 발생합니다. 보통은 세션이 시간 초과되고 있는 것이 원인입니다. 사용자는 계속하여 브라우저에서 다시 요구를 발생시키는 어떠한 작업을 수행합니다. 세션 시간 초과 값을 늘리거나 사용자를 초기 페이지로 이동하는 오류 메커니즘을 사용하여 작업을 다시 시작할 수 있습니다.
5919	요청한 동작은 잘못되었습니다.	일반적으로 암호화된 문자열을 CSP 페이지에서 Caché에 전달할 때, 데이터를 암호화할 때 사용한 키와 해독 키가 일치하지 않는 경우에 발생합니다. 이것은 사용자가 URL을 수동으로 변경하고, 암호화된 문자열이 Caché에서 생성된 후, 다음 HTTP 메시지 Caché에게 반송되는 사이에 문자열 값을 어떻게든 변경되면 발생합니다.

5920	네임 스페이스 % 1 이 CSP 페이지를 실행해야합니다.	각 CSP 응용 프로그램은 Caché 특정 네임 스페이스와 제휴하고 있습니다. 예를 들면, /csp / samples 이 SAMPLES 네임 스페이스에 연결된에도 불구하고, USER 네임 스페이스 /csp / samples / loop.csp 에서 페이지를 컴파일하려고 할 때 등에이 오류가 발생합니다.
5921	CSP 응용 프로그램 % 1 수행할 대상 네임 스페이스를 지정해야합니다.	CSP 응용 프로그램 구성에 네임 스페이스가 지정되지 않은 경우에 발생합니다. 시스템 관리 포털에서 네임 스페이스를 지정하지 않고 CSP 응용 프로그램을 만들 수 없기 때문에 일반적으로 이 오류는 CPF 파일을 수동으로 부정하게 변경되는 것을 나타냅니다.
5922	응답 대기 시간 초과입니다.	통신중인 HTTP 서버의 응답을 기다리는 % Net.HttpRequest 개체가 시간 초과되면 발생합니다.
5923	% 1 회 리디렉션되었습니다. 리디렉션 입니다.	1 개의 페이지에 5 개 이상의 리디렉션을 검출되면 발생합니다. 이 경우 컴파일러는 루프가 존재하는 것으로 간주합니다. CSP 페이지에서 다른 페이지로 이동하기 위해 ServerSideRedirect 를 사용하는 경우 A.csp 페이지 B.csp 리디렉션되어, 그곳에서 A.csp 리디렉션되는 경우 루프가 발생합니다.
5924	오류가 발생했습니다 지정 오류 페이지를 표시할 수 없습니다 - Web 마스터에게 통지합니다.	CSP 페이지에서 실행하는 동안 오류가 발생한 경우, CSP 엔진은 사용자가 지정 오류 페이지로 리디렉션됩니다. 이 오류 페이지에서는 유연하게 오류를 처리할 수 있습니다. 그러나 사용자 지정 오류 페이지가 존재하지 않거나 해당 오류 페이지를 생성하는 과정에서 오류가 발생했을 경우, CSP 엔진 BACK ^ % ETN 를 사용하여 문제가 발생했다는 사실을 기록하고 이 오류 메시지를 표시합니다. 사용자가 생성하는 오류 페이지에 버그가 있다고 프로덕션 시스템에서 이 오류가 발생할 수 있기 때문에이 메시지가 고의로 모호한

		표현이되어 있습니다. 이 오류를 해결하려면 먼저 CSP 응용 프로그램에서 지정된 오류 페이지가 있는지 확인한 다음 해당 오류 페이지에 버그가 있는지 확인합니다. " 부록 FAQ "이 오류에 관한 질문도 참조하십시오.
5925	<SCRIPT LANGUAGE=Cache> 태그 줄 번호 % 1, RUNAT 또는 METHOD 의 특성 중 하나가 없습니다.	이 코드를 실행 시기를 CSP 컴파일러에 지시하는 데 필요한 <i>runat</i> 특성 또는 새 메서드를 만들려면 <i>method</i> 메소드가< <b>script</b> <b>language="Cache"</b> >태그 않은 경우에 발생합니다.
5926	HTTP 헤더가 들어 플래시 되었기 때문에, 전송할 수 없습니다.	데이터가 브라우저에 기록된 후에 서버 쪽 리디렉션 기능을 사용하려고하면 발생합니다.% <i>response.ServerSideRedirect</i> 기능을 사용하여 다른 페이지로 리디렉션하는 경우 브라우저에 데이터가 반환되기 전에 실행해야합니다. 즉, 일반적으로 페이지의 <b>OnPreHTTP ()</b> 메소드 내에서 실행해야합니다.
5927	페이지 % 1 클래스 이름에 이미로드되어있는 클래스 % 2 가 충돌하여이 페이지를 로드할 수 없습니다.	다른 응용 프로그램에서 동일한 네임 스페이스에 같은 이름의 2 개의 CSP 파일을 설정하면 발생합니다. 예를 들어 USER 네임 스페이스에/ <b>test</b> 와 / <b>another test</b> 라는 2 개의 CSP 응용 프로그램이 있으며 각각은 Caché 서버의 다른 디렉토리에 존재하며, 각각의 디렉토리에 <b>test.csp</b> 파일이 있다고합니다. <b>autocompile</b> 을 선택하고 URL / <b>test</b> / <b>test.csp</b> 를 입력하면, CSP 컴파일러는이 페이지를 클래스 <b>csp.test</b> 컴파일합니다. URL/ <b>another test</b> / <b>test.csp</b> 를 입력하면 컴파일러는 클래스 <b>csp.test</b> 을 창조하기 위하여이 페이지를로드하려고하지만 다른 응용 프로그램에서 이미 존재하기 때문에이 오류가 발생합니다. 오류가 발생하지 않으면 요청이 발생할 때마다 전체 페이지를 다시 컴파일하는 등 낮은 기능의 동작을합니다. 같은 네임 스페이스에서 동일한 파일 이름을 사용하는 것을 피하거나, CSP 응용 프로그램에 정의된 패키지를 변경합니다. 이

		<p>패키지는 기본적으로 <b>csp</b>로 설정됩니다. 예를 들어, 패키지 이름 <b>package</b>를 사용하도록 / <b>another</b><b>test</b>을 변경합니다. 이제 <b>test.csp</b> 컴파일하면 <b>csp.test</b>를 사용하는 다른 응용 프로그램과 충돌하지 않는 클래스 이름 <b>package.test</b>가 생성됩니다.</p>
5931	페이지가 표시되기 전에 OnPreHTTP ()이 메소드를 호출하거나 값을 설정할 수 있습니다.	어떤 데이터가 브라우저에 표시되기 전에 몇 가지 매개 변수를 변경할 수 있도록 페이지 <b>OnPreHTTP()</b> 메서드 호출되고 있는 함수를 직접 호출하는 경우 발생합니다. 이 문제를 해결하려면 이를 호출 <b>OnPreHTTP</b> 메서드로 이동합니다.
5932	Web 서버에서 CSP 게이트웨이 버전에서 동작하지 않습니다.	사용하는 CSP 게이트웨이 동작을 지원하지 않는 버전이면 발생합니다. 이 기능을 사용하지 않거나, CSP 게이트웨이 버전을 새로운 것으로 업그레이드하십시오.
5933	CSP 서버에 내부 오류가 있습니다 : % 1	CSP 엔진에서 예기치 않은 오류 상황이 발생했을 경우에 발생합니다. 인터 시스템즈 기술 지원 서비스에 문의하십시오.
5954	CSP 페이지 잠금에 실패했습니다.	CSP 페이지가 자동으로 컴파일할 때, 2 개의 작업이 동시에 같은 페이지를 컴파일하지 않도록 먼저 이것을 잠금니다. 한쪽 작업이 60 초 이내에 잠금이 해제됩니다. 없는 사람에게 어떠한 원인으로 컴파일에 실패했다고 간주하고 이 오류 메시지가 표시됩니다. Caché 스튜디오에서 해당 페이지를 다시 컴파일하여 오류가 발생하는지 여부를 확인합니다.
5955	CSPAppList 검색어 : Fetch ()에 잘못된 데이터가 있습니다.	CSP 응용 프로그램 목록을 결정하기 위한 질문이 잘못된 경우에 나타납니다. 이 오류는 실행중인 시스템에서는 발생하지 않습니다.
5956	CSP 응용 프로그램 % 2 % 1 딕렉토리가 존재하지 않습니다.	CSP 응용 프로그램에서 지정된 딕렉토리가 파일 시스템에 없는 경우 발생합니다.

5961	문자 % 1 을 변환할 수 없습니다.	브라우저에서 요청을 받을 때 발생합니다. 브라우저에서 보낸 정보가 Caché 현재의 기본 로케일로 변환하는 동안 오류가 발생했습니다. 변환을 디버깅하려면 브라우저에서 보낸 정보를 분리하여 테스트 프로그램에서 수동으로 문자 집합을 변환합니다.
5962	새 세션을 할당할 수 없습니다.	이 세션 ID 새 슬롯이 존재하지 않을 때 % <b>session.ForceNewSession ()</b> 를 호출하면 발생합니다.
5963	잘못된 SysLog 수준 : % 1	내부 로그 수준을 설정할 때 수준 허용 오차로 설정되어있는 경우 발생합니다.