
Anyframe Oden User Guide

[Anyframe Documentation](#)

License

Copyright © 2009 SAMSUNG SDS, Co., Ltd.
All rights reserved.

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License.

You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

Every effort has been made to ensure that the information in this document is accurate. SAMSUNG SDS is not responsible for typographical errors.

SAMSUNG SDS Co., Ltd.
159-9, Gumi-Dong, Bundang-Gu,
Seongnam-Si, Kyunggi-Do, Korea 463-810
anyframe@samsung.com

SAMSUNG SDS, SAMSUNG SDS logo, Anyframe Java, Anyframe Java logo and all Anyframe Java-based trademarks are trademarks of SAMSUNG SDS Co., Ltd., registered in the Republic of Korea.

Java and all Java-based trademarks are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. Other company, product, and service names may be trademarks or service of others.

Technical Support

For more information, check "Anyframe Open Source Community (<http://www.anyframejava.org>)".
For technical support regarding using Anyframe Java products under open source license, visit "Anyframe Open Source Community Forum (<http://www.anyframejava.org/forum>)".

Contents

OVERVIEW	1
INTRODUCTION	1
KEY FEATURES	2
CONCEPTS	4
ARCHITECTURE	4
SERVERS AND AGENTS	6
SERVERS	6
AGENTS	7
DEPLOYING ITEMS	7
POLICIES	7
TASKS	8
BACKUP AND RESTORE	8
SNAPSHOT PLANS	8
SNAPSHOT AND ROLLBACK	9
USER INTERFACE	9
COMMAND LINE INTERFACE	9
GRAPHICAL USER INTERFACE	10
GETTING STARTED	11
SYSTEM REQUIREMENTS	11
INSTALLING ANYFRAME ODEN	11
SERVER AND AGENTS	11
ECLIPSE PLUG-IN	15
WORKING WITH COMMAND LINE	17
INTRODUCTION TO ODEN SHELL	17
ODEN SHELL COMMANDS	17
POLICY COMMAND	18
TASK COMMAND	21
HISTORY COMMAND	22
SNAPSHOT COMMAND	23
ROLLBACK COMMAND	25
VIEWING HELP	25
WORKING WITH ECLIPSE PLUG-IN	26

INTRODUCTION TO ODEN ECLIPSE PLUG-IN	26
ODEN EXPLORER VIEW	27
SERVERS	28
BUILD REPOSITORIES	29
ODEN POLICY AND TASK EDITOR	31
POLICIES TAB	31
TASKS TAB	34
ODEN SNAPSHOT VIEW	34
MANAGING SNAPSHOT PLANS	35
MANAGING SNAPSHOTS	37
ROLLBACK WITH SNAPSHOTS	37
ODEN DEPLOYMENT HISTORY VIEW	37
SEARCHING DEPLOYMENT HISTORY	38
ADVANCED SEARCH	38

Overview

Anyframe Open Deployment ENvironment (이하 Oden)¹은 CI²환경을 통해 빌드된 어플리케이션 컴포넌트 및 각종 설정파일, 웹파일 등을 원하는 배포대상서버에 편리하게 배포할 수 있도록 하는 배포관리 툴이다.

본 장에서는 Oden의 개발 배경 및 주요 특징에 대한 간략한 소개를 제공한다.

Introduction

최근의 일반적인 중대형 개발 프로젝트의 프로세스는 요구정의로부터 시작하여 분석 및 아키텍처정의, 설계, 개발, 이행 등으로 진행되는 것이 최근의 추세이며, 여러 벤더들은 이러한 각 공정 단계에 특화된 다양한 툴을 제공하고 있다.

그러나 분석/아키텍처정의/설계 단계에서 활용할 수 있는 다양한 툴과는 달리, 개발단계 이후 활용할 수 있는 툴의 범위는 다소 부족한 듯 하다. 특히, 개발한 어플리케이션 컴포넌트들을 개발서버 및 테스트서버 혹은 그 너머의 운영서버 등에 배포할 수 있는 전문화된 배포관리 툴에 대한 선택의 폭은 매우 작은 것이 현실이다.

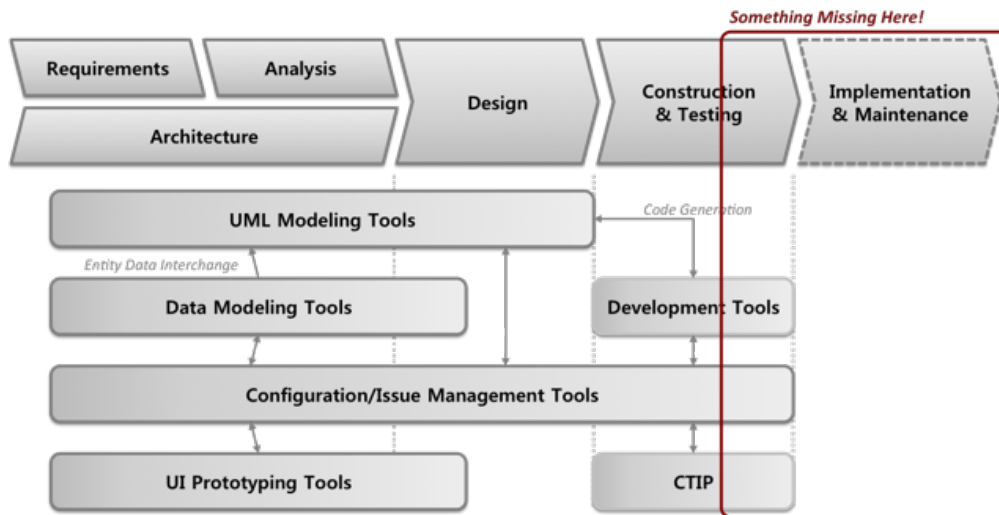


Figure 1. Something is Missing

¹ Oden은 OSGi, Eclipse 등 Java 기술을 활용하고 있으며, 손쉽게 확장이 가능한 개방형 아키텍처를 지향한다. 더불어 각종 UNIX, Mac OS X, Microsoft Windows 등 다양한 플랫폼 환경을 지원한다.

² Continuous Integration: 빌드, 테스트, 배포 등의 일련의 절차를 통합/자동화함으로써 소프트웨어 딜리버리의 시간을 단축시키는 기법 및 이를 가능하게 하는 것

이러한 전문적인 배포관리툴의 부재로 인해, 현장 프로젝트에서는 몇가지 어려움에 직면하게 된다.

첫째, 자동화 및 정형화된 배포관리가 이루어지지 않아 프로젝트 진행 및 운영 시 추가적인 리소스가 필요하게 된다.

- 기존에는 수작업 또는 CI엔진에 의한 복사 등으로 배포를 실시하였는데, 이를 관리하기 위해 QAO³혹은 SA⁴에 의한 배포관리가 전문적으로 이루어져야 했음
- 배포대상서버가 여러대일 경우, 해당 작업을 단순반복하여 처리해야 하므로 번거로운 작업을 수행해야 함

둘째, 배포 시 고려할 수 있는 다양한 배포 방법에 일일이 대응하기가 어렵게 된다.

- 전문적인 배포관리 툴이 없다면 전체 배포, 원하는 것만 배포, 변경된 사항만 배포 등 현장에서 요구하는 다양한 배포 방식에 일일이 대응하기 어려움
- 특히, 개발서버에서는 대개 변경된 사항만 배포되면 족함에도 불구하고, 일일이 비교하는 것이 번거롭기 때문에 전체를 한꺼번에 배포할 경우가 많음
- 실제로 일일이 비교하여 배포하는 경우라도, 누락되는 것이 있어 결국 배포에 실패하게 되는 경우가 발생함

셋째, 표준화 및 정형화된 프로세스에 기반한 개발 및 이행단계 진행이 어렵게 된다.

- 프로세스화된 배포 환경의 부재는 매번 배포시마다 업무의 혼란 및 리소스의 낭비 여지를 제공하게 됨
- 이를 위해 배포 정책 설정, 스냅샷/롤백, 로그분석, 스케줄링/배치, 워크플로우 적용 등 다양한 기능들이 필요함

Oden은 이러한 어려움을 극복하기 위한 자동화된 배포관리 환경을 제공한다.

Key Features

Oden은 다음과 같은 주요 특징을 지닌다.

첫째, 현장 프로젝트의 다양한 상황에 대응하기 위한 개방적이고 유연한 구조를 지향한다.

- 현장의 추가기능 요구에 유연하게 대응할 수 있도록 플러그인 형태(OSGi Bundle)로 기능을 추가할 수 있는 플랫폼을 활용하였음
- 배포관리서버를 위한 별도의 하드웨어를 두지 않아도 되는 유연한 구조를 지향함

³ Quality Assurance Officer: 품질관리자

⁴ Software Architect: 소프트웨어 아키텍트

둘째, 다양한 형태의 배포 방법을 지원한다.

- 배포 가능한 배포 대상에 대한 일괄 배포 (full-deployment)
- 배포 가능한 배포 대상 중 원하는 것만 선택적으로 배포 (selective-deployment)
- 배포 가능한 배포 대상 중 변경된 것만 배포 (incremental-deployment)

셋째, 배포관련 다양한 부가기능 및 가이드를 제공한다.

- 로그를 통한 이력 조회 및 결과 추적 기능 제공
- 스냅샷/롤백을 통한 배포 되돌리기 기능 제공
- 스케줄링, 워크플로우 등을 위한 외부 시스템과의 연계 고려
- CI엔진에서 빌드된 내용으로부터 배포 실시 및 Incremental Build 등 CTIP 환경 가이드 제공

넷째, 다양한 배포환경에 적용할 수 있도록 안정적인 성능을 보장한다.

- 대량, 대용량 배포물에 대한 안정적인 배포실행 제공⁵
- 물리적 디스크 I/O를 최대한 줄여 속도 확보

다섯째, 간단한 조작을 통한 손쉬운 배포 방식을 제공한다.

- Oden Shell에서 제공하는 CLI⁶ 환경을 통한 배포 실행
- Eclipse UI에서 제공하는 GUI 환경을 통해 CLI 환경의 다양한 조작을 일관된 프로세스 흐름을 통해 동일하게 진행
- CLI, GUI 환경에서 각각의 환경에 맞는 도움말 제공

⁵ Anyframe Oden 개발팀 내부의 자체적인 성능검사 결과, 대량(25,000여 개의 컴포넌트) 및 대용량(6GB 정도의 용량) 배포환경에서 안정적인 배포결과를 보여주었다.

⁶ Command Line Interface: 커맨드라인 인터페이스. 일반적으로 터미널 등을 통한 명령프롬프트 기반의 사용자 작용에 기반한다. 대표적인 CLI 환경으로는 UNIX Shell 환경을 예로 들 수 있다.

Concepts

본 장에서는 Oden에 대한 보다 상세한 이해를 돕기 위해, Oden에서 지향하는 아키텍처의 형태 및 주요 구성요소에 대한 소개를 제공한다.

Architecture

Oden을 활용한 배포관리 환경은 크게 다음과 같은 구성요소로 이루어진다.

- **Server & Agent**
배포관련 각종 작업을 수행하는 핵심모듈로, Server는 빌드리파지토리 (Build Repository) 혹은 배포대상서버 (Target Server) 등에 위치시킬 수 있으며, Agent는 배포대상서버에 설치되며, CLI 기반 UI를 제공함
- **Build Repository**
CI 환경에 의해 빌드된 각종 컴포넌트들이 저장된 장소로, 배포대상서버의 디렉토리 구조와 동일하게 구성되어야 함
- **GUI Tool**
Eclipse Plug-in 등을 활용하여 Oden 관련 각종 배포작업을 수행하는 GUI 환경

Oden을 활용하기 위해 Eclipse Plug-in을 사용할 경우, Eclipse의 Plug-in 기반 아키텍처의 장점에 의해, 모델링-코드생성-빌드-테스트-배포 및 형상/이슈관리를 유기적으로 통합시킬 수 있다. 이러한 일련의 흐름을 예를 들어 설명하면 다음과 같다.

- (1) 설계모델링을 통해 Eclipse 프로젝트 및 소스코드를 생성한다.
- (2) 생성된 코드를 바탕으로 필요한 비즈니스로직을 추가하여 프로그램을 작성한다.
- (3) 이러한 개발을 진행할 때에는 테스트케이스를 함께 생성하여 진행함으로써 개발생산성 및 유지보수성을 증대시킨다.
- (4) CI 엔진과 연계하여 빌드 시 테스트 수행을 포함하여 통합빌드를 진행한다.
- (5) 배포관리툴과 연계하여 원하는 배포대상서버에 배포를 실시한다.
- (6) 이러한 일련의 과정을 이슈관리 시스템 및 형상관리 시스템과 연계시켜 체계적으로 관리한다.

다음 그림은 이러한 유기적인 통합의 모습을 개념적으로 나타낸 것이다.

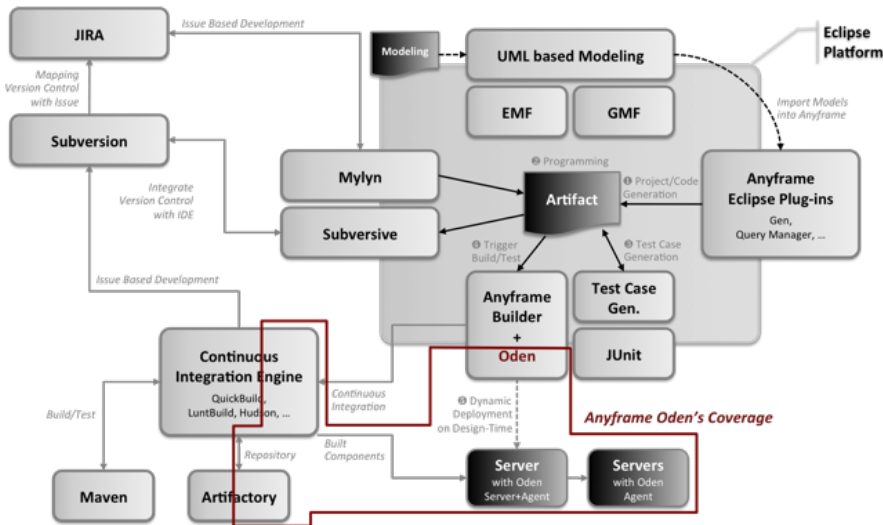


Figure 2. Oden Integrated Architecture powered by Eclipse

Oden을 활용한 배포환경 구성 시에는 Oden에서 제공하는 Server, Agent, Eclipse Plug-in 등을 각각의 위치에 설치하여야 하며, 이때 Server의 위치는 빌드리파지토리나 배포대상서버, 어느 곳이든 상관없지만, Agent의 경우에는 반드시 배포대상서버에 위치시키도록 한다. 설치 관련 자세한 사항은 "Installing Anyframe Oden"을 참고하도록 한다.

Note
Server는 빌드리파지토리 혹은 배포대상서버에 위치시키고, Agent는 배포대상서버에 위치시킨다.

Oden을 활용하여 배포대상물을 원하는 배포대상서버에 배포하고, 로그를 통해 결과를 조회하며, 원하지 않는 경우 스냅샷을 뜬 시점으로 롤백할 수 있다. 이러한 일련의 내용을 각 구간별로 설명하면 다음과 같다.

Eclipse Plug-in UI를 통한 배포관리

- 배포정책 (Policy; Full/Selective/Incremental) 설정
- 배포실행 (Task)
- 로그를 통한 결과 확인/재실행
- 스냅샷 관리 및 실행
- 롤백 실행

Console (CLI)을 통한 배포관리

- Eclipse Plug-in UI (GUI)와 동일한 기능 수행

빌드리파지토리에서 배포대상물을 가져옴

- 각종 배포대상물 (클래스/JAR/WAR, 웹리소스 등)에 대한 변경사항 감지
- CI를 활용한 빌드 시 Incremental Build 방법을 적용할 것을 가이드함

Oden Server가 설치된 서버가 "HOST" 역할을 수행함

- 배포정책 (Policy) 저장
- 배포실행 (Task) 및 로그생성
- 바뀐 부분을 감지하여 배포할 파일만 Agent에게 전달
- 스냅샷 및 롤백 수행
- Server를 CI서버/빌드리파지토리에 위치시키거나 별도의 장비, 혹은 배포대상서버들 중 하나에 설치하는 것이 가능함

Oden Agent만 설치된 배포대상 서버들은 "GUEST" 역할을 수행함

- Server로부터 전달받은 배포대상물들을 지정된 위치에 배포함

다음 그림은 이러한 배포 흐름을 개념적으로 설명한 것이다.

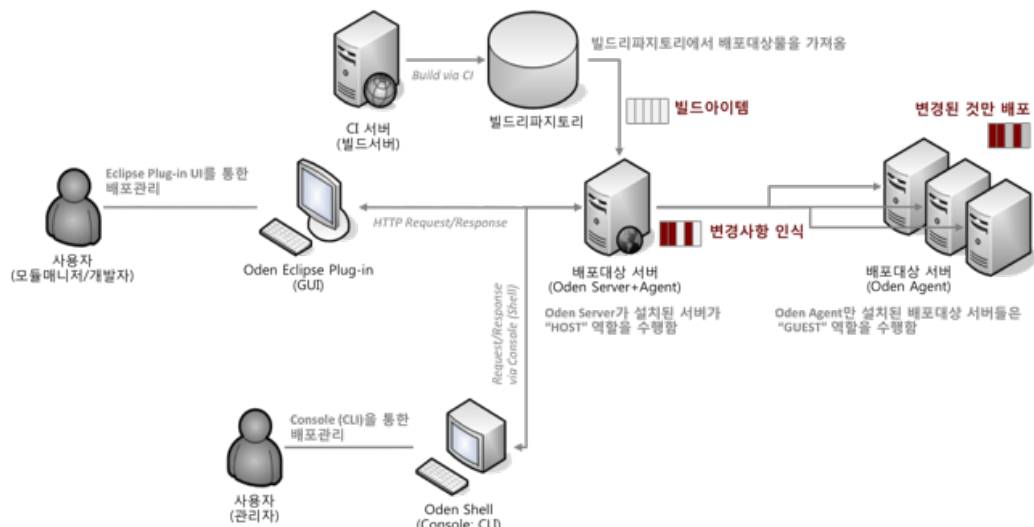


Figure 3. Oden Deployment Flow

Servers and Agents

Oden의 Server 및 Agent는 배포작업을 실질적으로 관할하고 수행하는 역할을 수행한다. Server 및 Agent는 배포할 것들을 확인, 감지, 목록화, 전송, 배포하는 일련의 작업들을 수행하며, 이러한 것들은 각각의 기능구성 및 설치 환경 등을 고려하여 개별 번들⁷로 구성되어 있다.

Servers

Oden Server는 각종 Oden Shell을 통한 셸 커맨드를 제공하며, 이를 통해 다양한 배포관련 명령을 수행할 수 있도록 한다. 또한, Eclipse Plug-in 등 외부 GUI 환경과의 연계를 위한 서비스도 제공한다.

⁷ Oden Server 및 Agent는 OSGi 표준사양을 따르는 개별 번들로 구성되어 있다.

이러한 Oden Server는 단일 어플리케이션 당 하나로 구성하는 것을 권장하며, 여러대의 배포대상서버⁸에 대한 배포를 하나의 Server 및 여러개의 Agent를 통해 수행하게 된다. 예를 들어, 'A'라는 업무도메인에 대한 어플리케이션을 3대의 배포대상서버에 배포하고, 'B'라는 업무도메인에 대한 어플리케이션을 2대의 배포대상서버에 배포한다면, 총 2개의 Server를 구성하는 것을 권장한다. 이는, 각 어플리케이션 당 관련된 빌드리파지토리만을 특정 Oden Server와 연결하는 것이 관리하기 편하기 때문이다.

$$\text{Number of Oden Servers} = \sum (\text{Number of Applications})$$

Agents

Oden Agent는 Oden Server로부터 전달받은 배포대상물을 지정된 위치에 배포(복사)하는 역할을 수행한다.

그러므로, 앞서 예를 든 'A' 및 'B'라는 업무도메인에 대한 배포대상서버가 각각 3대 및 2대이므로, Oden Agent는 업무도메인 'A'용 Oden Server에 대해서 3개를, 업무도메인 'B'용 Oden Server에 대해서 2개를 각각의 배포대상서버에 구성한다.

$$\text{Number of Oden Agents} = \sum (\text{Number of Target Servers})$$

만약 배포대상서버가 물리적으로 중복된 장비에 위치해 있다하더라도, 구성하는 Agent의 개수는 변하지 않는다. Oden의 Server 및 Agent의 구성은 논리적인 구성에 입각하여 구성하는 것을 권장한다.

Note

Server는 단일 어플리케이션 당 하나로 구성하는 것을 권장하며, Agent는 해당 어플리케이션의 배포대상서버 당 하나를 구성한다.

Deploying Items

Oden은 배포작업의 효율성을 위해 배포정책(Policy) 및 배포작업(Task)에 의해 배포를 실시한다.

Policies

특정 아이템을 배포하겠다는 내용을 “배포정책”으로 서버에 저장하여, 개별 사용자가 정의한 배포정책을 다수의 사용자가 공유하여 활용할 수 있다.

배포정책에 대한 보다 자세한 내용은 "Policy Command" 및 "Oden Policy and Task Editor" 등을 참고한다.

⁸ Oden은 물리적인 하드웨어가 아닌 웹어플리케이션서버(WAS) 또는 서블릿컨테이너 등의 인스턴스 하나를 배포대상서버 하나로 간주한다.

Tasks

하나 이상의 배포정책을 묶어서 “배포작업”으로 설정한 후, 해당 배포작업을 실행시키면, 배포작업 내에 정의된 배포정책에 의해 배포대상물들이 배포된다.

배포작업에 대한 보다 자세한 내용은 "Task Command" 및 "Oden Policy and Task Editor" 등을 참고한다.

배포를 수행하는데 번거롭게 여겨질 수도 있는 배포정책(Policy)와 배포작업(Task)의 다단계 구성을 한 이유는 다음과 같다.

- 한번 작성한 배포정책을 여러사람이 여러번 재사용할 수 있도록 하게 함
- 배포정책 자체를 꼭 필요한 범위만으로 설정할 수 있는 모듈 구성으로 만들어서 빈번한 배포 시에 보다 신속한 배포작업을 가능하게 함
- 여러개의 배포정책을 하나로 묶어서 한번에 실행하는 배포작업을 구성할 수 있도록 하게 함

이상의 Policy와 Task 간의 관계를 그림으로 나타내면 다음과 같다.



Figure 4. Policy and Task

Backup and Restore

Oden은 특정 시점의 배포형상을 별도로 보관하고 이를 통해 배포형상을 되돌리기 위한 롤백 기능을 지원한다.

Snapshot Plans

배포대상서버의 특정 위치를 지정하여 스냅샷을 뜨도록 스냅샷정책을 설정한다.

스냅샷정책은 특정 위치에 배포된 파일 및 디렉토리 등을 백업하기 위한 설정이며, 이에 대한 보다 자세한 내용은 "Snapshot Command" 및 "Oden Snapshot View" 등을 참고한다.

Snapshot and Rollback

정의한 스냅샷정책에 의해 실제 스냅샷을 생성함으로써 백업을 수행하고, 이를 바탕으로한 롤백을 통해 복구를 수행한다.

스냅샷은 스냅샷정책에서 정의된 위치의 파일 및 디렉토리 등을 압축⁹한 파일이며, 롤백은 이 파일을 스냅샷정책을 참고하여 다시 원래의 위치에 풀어놓는 방식이다. 스냅샷 및 롤백에 대한 보다 자세한 내용은 "Snapshot Command" 및 "Rollback Command", "Oden Snapshot View" 등을 참고한다.

스냅샷정책(Snapshot Plan)과 스냅샷(Snapshot) 역시 한번 정의한 스냅샷정책에 의해 여러번의 스냅샷을 뜰 수 있도록 하기 위해 단일작업이 아닌 다단계 작업에 의해 수행된다. 이러한 관계를 그림으로 나타내면 다음과 같다.

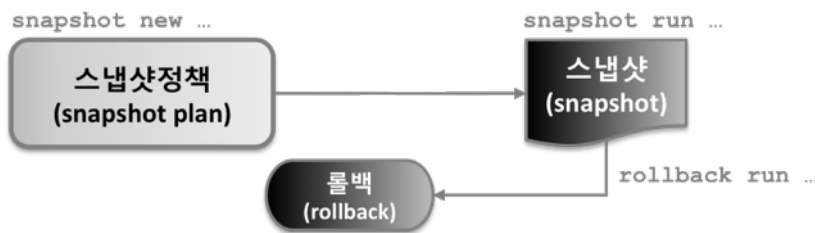


Figure 5. Snapshot Plan, Snapshot and Rollback

User Interface

Oden은 CLI 및 GUI 환경 모두에서 작업이 가능하며, 사용자는 두 환경 모두에서 동일한 배포 결과를 얻을 수 있다.

Command Line Interface

Oden 명령어를 실행시키기 위해 Oden에서는 자체 Shell 환경을 제공한다. Oden Shell을 통해 사용자는 Oden에서 제공하는 기능들을 이용할 수 있다.

보다 자세한 사항은 "Working with Command Line"을 참고한다.

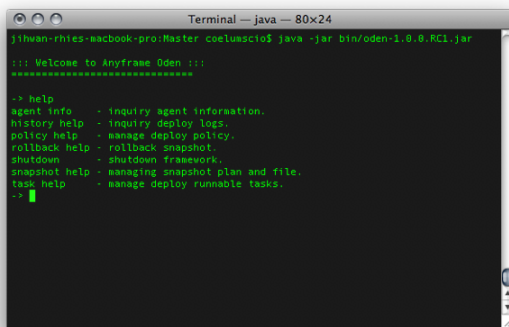


Figure 6. Oden Command Line Interface: Oden Shell

⁹ 스냅샷은 확장자가 없는 ZIP 압축 파일이다.

Graphical User Interface

Oden Eclipse Plug-in은 배포관리를 위한 Oden Server와 연계 하여 배포관리를 지원하는 플러그인으로, 배포 용이성 및 높은 신뢰성을 기대하게 해 준다.

보다 자세한 사항은 "Working with Eclipse Plug-in"을 참고한다.

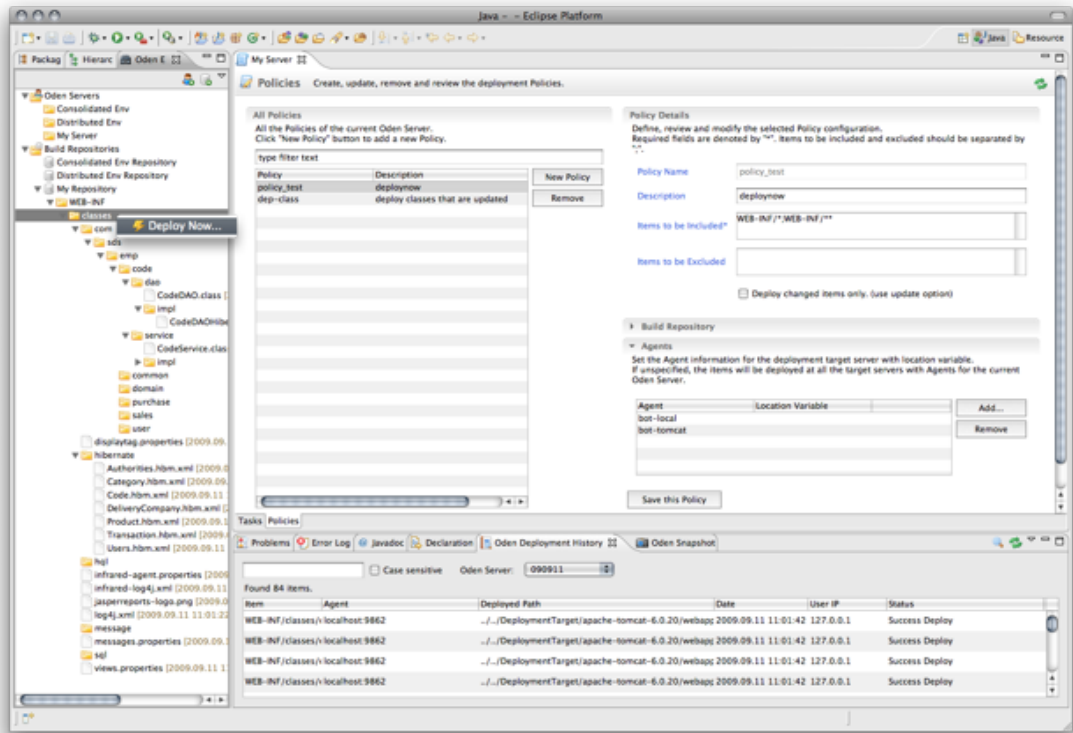


Figure 7. Oden Graphical User Interface: Eclipse Plug-in

Getting Started

본 장에서는 Oden을 사용하기 위한 설치요구사항과 설치 및 설정에 대한 소개를 제공한다.

System Requirements

Anyframe Oden을 설치하기 위해서는 다음의 요구사항을 충족해야 한다.

Oden Server 및 Agent 설치요구사항

- Java Runtime Environment 1.5.0 or above
- 10MB Disk

Oden Eclipse Plug-in 설치요구사항

- Java Runtime Environment 1.5.0 or above
- Eclipse IDE for Java EE Developers (Ganymede/Galileo) or equivalent
- Anyframe Common 3.0.0 or above
- 기타 하드웨어 및 소프트웨어 요구사항은 위 Eclipse 버전별 요구사항을 따름

Note

Anyframe Java 개발을 원활하게 진행하기 위해 “Anyframe Open Source Community¹⁰”에서 배포하는 Eclipse 배포본의 사용을 권장한다.

Installing Anyframe Oden

Server and Agents

Oden은 Server와 Agent로 구성되어 있다. Agent는 파일이 배포될 서버인 배포대상서버 (Target Server)¹¹에 설치되어 Oden Server에서 지시하는 명령에 따라 파일을 배포한다. Oden Server는 여러개의 Agent를 관리하며 사용자는 Shell이나 Eclipse Plug-in과 같은 UI환경을 통해 Oden Server에 명령을 내리게 된다. 사용자는 Agent에 직접 명령을 내릴 수 없으며, Oden Server를 통해 Agent에

¹⁰ <http://www.anyframejava.org/>

¹¹ 이 문서에서는 파일이 배포될 서버를 Target Server라 칭한다.

명령을 내려야 한다. Oden Server는 기본적으로 Agent를 하나 내장하고 있으므로, Oden Server가 설치된 장비에는 별도의 Agent를 설치할 필요가 없다.

Installing Server and Agents

배포될 파일이 존재하는 빌드리퍼지토리 (Build Repository)¹²와 파일을 배포할 서버인 Target Server가 있을 때, Oden Server를 Build Repository에 설치할 수도 있고 Target Server에 설치할 수도 있다. Target Server에 Oden Server를 설치하게 되면 Oden Agent를 따로 설치할 필요없이 Oden Server에 내장되어 있는 Oden Agent를 이용할 수 있다. 다만, Build Repository는 물리적으로 다른 장비에 위치하고 있을 경우에는 둘 간의 통신을 위해 FTP를 사용해야 한다. Build Repository에 Oden Server를 설치하게 되면, Target Server에 Oden Agent를 별도로 설치해야 하나 FTP를 사용하여 Build Repository에 접근하지 않아도 되므로 속도가 빠르다.

Oden Server의 압축을 풀면 아래와 같은 구조로 되어 있다.

```
[Oden Server 설치 디렉토리]
|
+---- bin
| |
| +---- oden-x.x.x.jar    // 실행할 바이너리 파일
|
+---- bundle    // Oden 실행시 필요한 library 파일들이 들어 있음.
|
+---- conf
|
|   +---- oden.ini    // Oden Server 설정파일
|
|   +---- config.xml  // Oden Agent 및 기타 Oden 설정이 들어 있는 파일
```

Oden Agent는 아래와 같은 구조로 되어 있다.

```
[Oden Agent 설치 디렉토리]
|
+---- bin
| |
| +---- oden-x.x.x.jar    // 실행할 바이너리 파일
|
+---- bundle    // Oden 실행시 필요한 library 파일들이 들어 있음.
```

¹² 이 문서에서는 배포할 파일이 존재하는 장소를 Build Repository라 칭한다. 일반적으로 빌드가 끝나고 class나 lib과 같은 바이너리 파일이 존재하는 장소가 될 것이다.

Oden Server와 Agent의 실행 및 설정에 관한 사항은 다음을 참고하도록 한다.

Configuring Server and Agents

앞에서 언급했듯이 Oden Server를 Build Repository에 설치할 수도 있고 Target Server에 설치할 수도 있다. 여기서는 우선 Build Repository에 Oden Server를 설치하고 Target Server에 Oden Agent를 설치했다고 가정하고 Oden 설정을 하는 방법을 설명하도록 하겠다. Oden Server에서 Oden Agent를 인식시키게 하기 위해서는 conf/config.xml에 Oden Agent의 정보를 입력해 주어야 한다. conf/config.xml을 열어 Oden Agent가 제대로 등록이 되어 있는지 확인한다.

```
...
<agents>
  <agent name="agent0">
    <address host="177.11.22.33" port="9862"/>
    <default-location value="C:/tomcat/webapps/anyframe-app"/>
  </agent>
</agents>
...
```

Oden Agent가 설치된 Target Server의 주소(여기서는 177.11.22.33)를 <address .../> 란에 넣어 주었다. Agent의 기본 포트는 9862이며 실행시 -port옵션을 통해 변경할 수 있다. Oden Agent에 배포할 때 따로 location명을 지정하지 않으면 default-location을 사용한다. 여기서는 Target Server에 있는 application의 root 디렉터리를 default-location으로 지정하였다. config.xml에 Agent정보를 추가하였으면, 우선 Oden을 사용할 준비는 끝났다. (config.xml은 수정 후 재부팅하지 않아도 된다.) 이번에는 Build Repository에 Oden Server를 설치하고 Target Server에 Oden Agent를 설치하는 방법을 설명하도록 하겠다. Target Server에 Oden Server를 설치하였기 때문에 별도의 Agent를 설치하지 않고 Oden Server에 내장되어 있는 Agent를 통해 파일을 배포할 것이다. 내장된 Agent라 하더라도 Oden Server에서 Agent를 인식하게 하기 위해서는 conf/config.xml에 Agent의 정보를 입력해 주어야 한다. Build Repository는 떨어져 있으므로 Oden 명령어를 실행할 때 FTP 프로토콜을 이용하여 Build Repository에 접근하여야 한다. conf/config.xml을 열어 Oden Agent가 제대로 등록이 되어 있는지 확인한다.

```
...
<agents>
  <agent name="agent0">
    <address host="localhost" port="9862"/>
    <default-location value="C:/tomcat/webapps/anyframe-app"/>
  </agent>
</agents>
...
```

여기서는 Oden Server에 내장된 Agent를 사용할 것이므로 address항목에 localhost를 넣어 주었다. Agent의 기본 포트는 9862이며 conf/oden.ini나 실행시 -port옵션에서 변경할 수 있다. Oden에서

지금 등록한 Agent에 배포할 때 따로 location명을 지정하지 않으면 default-location을 사용한다. 여기서는 Target Server(Oden Server가 설치되어 있는 PC)에 존재하는 application의 root 디렉터리를 default-location으로 지정하였다. config.xml은 수정후 재부팅하지 않아도 된다. config.xml에 Agent정보를 추가하였으면, 우선 Oden을 사용할 준비는 끝났다. oden.ini 와 config.xml을 통해 Oden Server의 설정을 변경할 수 있다. Oden Agent는 oden.ini 파일이 없으므로 Oden Agent의 포트 변경을 위해서는 구동시 -port <포트번호> 옵션을 사용하여 구동시키면 된다.

Note

oden.ini를 통해 Oden Server자체의 설정을 변경할 수 있다. 변경 내용을 적용하기 위해서는 서버를 재구동 시켜야 한다.

oden.ini 파일에서 정의하는 정보는 다음과 같다.

- **bundle.start**
Oden Server 구동과 동시에 구동 될 library 목록. 수정할 필요 없음.
- **log.level**
log파일에 뿌려질 로그 레벨(1 = error, 2 = warning, 3 = info, 4 = debug). 로그파일은 meta폴더에서 확인할 수 있다. Oden에 문제가 생겼을 시 디버깅하기 위한 용도로 사용되며, 일반적으로 수정할 필요는 없다.
- **http.port**
Telnet이 아닌 Eclipse Plug-in과 같은 UI와 통신하기 위한 포트.
- **shell.port**
Telnet으로 Oden Shell에 접근할 때 사용하는 포트.
- **agent.port**
Oden Server에 내장된 Agent에 접근하기 위한 포트.

Note

Oden Agent를 Oden Server에서 인식하게 하기 위해서는 config.xml에 Agent정보를 추가해 줘야 한다. Oden Server 실행과 동시에 구동되는 Oden Agent의 정보 역시 config.xml에 설정해 줘야 한다. config.xml의 값은 Oden Server의 재구동 없이 바로 적용된다. 기본 형태는 아래와 같다.

```
<oden>
  <agents>
    <agent name="">
      <address host="" port=""/>
      <default-location value=""/>
      <location name="" value=""/>
      <location /> ...
    </agent>
    <agent name="">
      <!-- 이 곳에 두번째 Agent의 정보를 설정한다. -->
    </agent>
  </agents>
</oden>
```

config.xml 파일에서 정의하는 내용은 다음과 같다.

- **agent name**
Agent를 지칭할 Unique한 값을 넣어준다. 이 이름은 Oden 명령어에서 사용될 것이다.
- **address**
host속성에는 Agent가 설치된 서버의 IP(Server에 내장된 Agent일 경우 localhost)를 입력한다.
port속성에는 Agent의 포트(기본값 9862. oden.ini 혹은 실행시 -port옵션에서 수정할 수 있다.) 값을 입력한다. 수정하지 않았다면 기본 값인 9862를 입력하여 준다.
- **default-location**
Agent로 파일을 전송할 때 파일이 배포될 경로로 절대 경로를 입력한다. Oden 명령어 실행시 Agent의 location명을 지정하지 않으면 default-location이 사용된다.
- **location**
default-location 외에 사용할 경로를 지정하는 태그이다. 역시 절대 경로를 입력해야 한다.
- Agent가 여러개 일 경우 동일한 방식으로 다른 서버에 설치되어 있는 Agent 정보를 추가해 주면 된다.

Starting Up Server and Agents

Oden Server 및 Agent의 설치 및 설정이 끝났으면, 아래의 명령으로 Oden Server를 실행시킬 수 있다.

```
cd <Oden Server가 설치된 경로>
java -jar bin/oden-x.x.x.jar
```

Oden Agent는 아래의 명령으로 실행시킬 수 있다.

```
cd <Oden Agent가 설치된 경로>
java -jar bin/oden-x.x.x.jar
```

Oden Server를 실행시키면 Oden Shell이 뜨며, help를 입력하면 Shell에서 사용가능한 명령들을 확인할 수 있다.

Eclipse Plug-in

Anyframe Oden은 Eclipse Plug-in으로도 제공된다. 다음은 Plug-in 설치에 대한 안내이다.

Installing Eclipse Plug-in

Eclipse의 Help > Install New Software를 통해서 Anyframe Update Site인 “<http://dev.anyframejava.org/update>”을 추가한다.

Important

Anyframe Oden은 Anyframe Common이 설치되어 있어야 설치가 가능하다.

Eclipse 3.4, Eclipse 3.5의 경우에는 Anyframe Oden에 Anyframe Common이 포함되어 있으므로 따로 설치하지 않아도 된다.

설치가 완료되면, Eclipse의 Help > About Elicpse를 통해서 확인할 수 있다.

Configuring Eclipse Plug-in

Server, Agent의 설정은 Working with Command Line을 참조한다.

Working with Command Line

Oden 명령어를 실행시키기 위해 Oden에서는 자체 Shell 환경을 제공한다. Oden Shell을 통해 사용자는 Oden에서 제공하는 기능들을 이용할 수 있다.

Introduction to Oden Shell

아래의 명령으로 Oden Server를 실행시키면 자동적으로 Oden Shell이 실행된다.¹³

```
cd <Oden Server가 설치된 경로>
java -jar bin/oden-x.x.x.jar [-port <포트번호>]
```

Oden Server는 기본적으로 하나의 Oden Agent를 내장하고 있으며, Oden Server실행과 동시에 내장되어 있는 Oden Agent도 실행되게 된다. -port 옵션의 포트번호는 Oden Agent의 포트번호를 의미하며 지정할 경우 conf/oden.ini 파일의 agent.port의 값을 대신하게 된다. 지정하지 않았을 경우 기본 값은 9862이다.

로컬뿐만 아니라 Telnet을 통해 원격으로 Oden Shell 에 접근할 수도 있다.

```
telnet <Oden Server IP> <Oden Shell Port>
```

Oden Shell Port의 기본 값은 9861이며 값을 수정하기 위해서는 Oden Server의 conf/oden.ini 파일의 shell.port값을 수정하면 된다. 수정 후 Oden Server를 재구동해야 한다. 로컬에 있는 Oden Server에 Telnet으로 접근하기 위해서는 IP에 localhost나 127.0.0.1과 같은 alias가 아닌 실제 Oden Server가 설치된 IP를 입력해 주어야 한다.

Oden Shell Commands

Oden Shell을 통해 Oden Shell Commands를 실행시킬 수 있다. 인자에 스페이스가 들어가 있으면 " "로 묶어줘야 한다.

¹³ Oden Agent는 입력 가능한 Shell환경을 제공하지 않는다.

Policy Command

Policy Command를 통해 Policy를 설정하고 테스트해 볼 수 있다. Oden에서 배포 작업을 수행하는 단위는 Task이지만, Task는 여러개의 Policy들로 구성되어 있기 때문에 Task를 설정하기 전에 Policy를 설정해야만 한다.¹⁴

policy add

Policy를 추가하기 위한 명령어. 동일한 이름을 가진 Policy가 있을 경우 덮어 쓴다.

```
policy add <policy-name>
  -r[epo] [file://<path> | [ftp://<host> <path> [<id> <password>]]
  -i[nclude] <wildcard-location> ... [-e[xclude] <wildcard-
location> ...]
  [-u[pdate]]
  -d[est] <agent-name>[/<location-variable-name>] ...
  [-desc <description>]
```

‘-r’은 Oden Server에서 접근할 Build Repository를 의미한다. Oden Server와 물리적으로 동일한 장비에 파일이 존재한다면 file 프로토콜을, 아니면 ftp프로토콜을 사용하면 된다.

‘-i’는 Build Repository에서 어떤 파일들을 배포할 것인지 지정하는 옵션이다. ‘*’같은 wildcard문자를 사용할 수 있으며, 하위경로까지 포함하려면 ‘**’를 사용해야 한다. 여러개의 인자가 있을 경우 ‘스페이스’로 구분한다. ‘-e’ 옵션을 지정할 경우 지정된 파일들은 배포대상에서 제외된다. 들어가는 인자의 형태는 ‘-i’ 옵션과 동일하다.

‘-u’ 옵션을 지정할 경우 Agent에 기 배포되어 있는 파일과 비교(날짜 비교) 하여 변경된 파일만 배포한다. jar파일일 경우 jar파일내에 포함되어 있는 파일 중 변경된 파일만 update하게 된다.

‘-d’는 Agent명과 Agent의 location을 지정한다. Agent와 location은 conf/config.xml에 이미 등록이 되어 있어야 한다. Agent와 location명은 / 로 구분하며, location을 지정하지 않았을 경우 conf/config.xml에 지정된 default-location에 배포하게 된다. 배포할 Agent가 여러개일 경우 스페이스로 구분한다.

다음은 실제 환경에서 참고하여 활용할 수 있는 다양한 예제들이다.

로컬 파일을 agent0로 배포

```
policy add p0 -r "file://C:/Build Repository/anyframe-app" -i ** -d
agent0 -desc "local files to agent0"
```

C:/Build Repository/anyframe-app의 모든 file(하위경로 포함)들을 agent0(conf/config.xml에 설정되어 있어야 함)의 default-location에 배포함. 옵션의 인자에 스페이스가 있을 경우 쌍따옴표로

¹⁴ Policy를 바로 실행할 수 없다. 해당 Policy를 가진 Task를 구성하여 실행해야 된다.

뭉어야 한다. **는 하위의 모든 경로를 포함함을 의미한다. 경로가 포함되면 그 안의 파일들도 배포가 되기 때문에 결국 **만 지정을 해주면 하위경로의 모든 파일을 배포함을 의미한다.

특정확장자를 제외한 파일만 배포

```
policy add p1 -r "file:///C:/Build Repository/anyframe-app" -i ** -e
**/*.svn -d agent0/temp
```

C:/Build Repository/anyframe-app의 모든 file(하위경로 포함)들중 svn확장자를 가진 파일이나 폴더를 제외하고 agent0의 temp에 배포함. temp는 conf/config.xml의 location에 정의되어 있어야 한다.

ftp로 특정확장자의 파일들을 여러 agent로 배포

```
policy add p2 -r ftp://172.16.70.77 /anyframe-app/images guest 1111 -i
**/*.jpg **/*.gif -d agent0/images agent1/images
```

172.16.70.77의 ftp에 guest/1111 계정으로 접근한 뒤 anyframe-app/images 하위의 모든 jpg, gif, png파일들을 agent0와 agent1의 images 경로로 배포함. agent0와 agent1, agent0의 images, agent1의 images는 conf/config.xml에 정의되어 있어야 한다.

ftp로 특정확장자의 파일들을 여러 agent로 배포2

```
policy add p2 -r ftp://172.16.70.77 /anyframe-app guest 1111 -i
images/**/*.jpg images/**/*.gif -d agent0 agent1
```

이번 예제는 config.xml에 images location을 등록하지 않고도 위의 예제와 동일한 결과를 얻을 수 있음을 보여준다. -i 옵션에서 지정된 경로는 그 경로 그대로 Agent로 배포된다. 예를들어 -i sample/**로 배포할 파일을 지정하면 Agent에 sample폴더 이하의 파일이 바로 배포되는 것이 아니라, sample폴더가 먼저 생성이 된 후 그 안에 파일들이 배포된다. conf/config.xml에 매번 location을 등록하는 것이 번거롭다면 이런식으로 사용하기 바란다.

변경된 파일만 배포

```
policy add p3 -r ftp://172.16.70.77 /anyframe-app/classes guest 1111 -i
** -u -d agent0/classes
```

172.16.70.77의 ftp에 guest/1111 계정으로 접근한 뒤 classes폴더의 모든 파일들(하위경로 포함)과 agent0의 classes의 파일들과 비교하여 날짜가 변경된 파일만 배포함. jar파일이 있을 경우 jar내의 변경된 파일들만 update함. conf/config.xml에 agent0(agent태그)와 classes(location태그)가 정의되어 있어야 한다.

Note

Ant를 이용하여 변경된 소스코드만 빌드하도록 할 수 있다. Ant의 javac 태스크는 기본적으로 소스 폴더와 빌드 폴더를 비교하여 class파일이 존재하지 않는 소스 코드에 대해서만 빌드를 수행하도록 되어 있다. 그래서 일반적으로 ant 스크립트 작성 javac를 수행하기 전에 빌드 폴더를 전부 삭제하여 Full Build를 수행한다. ant의 depend 태스크를 사용하면 빌드 폴더를 전부 삭제하지 않고 변경이 일어난 class 와 그것과 의존관계가 있는 class파일만 삭제할 수 있다. 이 후 javac 태스크에서는 존재하는 파일은 제외하고 존재하지 않는 파일(depend에 의해 삭제된 변경된 class 및 그것과 의존성이 있는 class)에 대해서만 빌드를 수행하므로 변경된 파일에 대해서만 빌드가 가능하다. 이렇게 변경된 파일과 그것과 의존성이 있는 파일에 대해서만 빌드하는 것을 Incremental Build라고 한다. 아래 예제는 depend 태스크를 이용하여 Incremental Build를 수행하는 스크립트를 보여준다.

```
<target name="inc-build">
  <depend srcdir="src/main/java" destdir="build/classes" closure="yes" />
  <javac srcdir="src/main/java" destdir="build/classes">
    <classpath>
      <path refid="compile.classpath" />
    </classpath>
  </javac>
</target>
```

depend 태스크에 의해 src/main/java와 build/classes를 비교하여 변경된 파일과 의존성있는 파일에 대해서만 class파일이 삭제된다. javac 태스크에 의해 삭제된 파일에 대해서만 빌드가 일어난다. 이 후 Oden Policy의 -u 옵션을 통해 변경된 바이너리 파일만 배포가 가능하다.

policy info

Policy의 정보를 조회하기 위한 명령어. policy-name을 지정하지 않았을 경우 등록된 모든 Policy의 정보를 보여준다.

```
policy info [<policy-name>]
```

policy del

Policy 삭제 명령어. policy-name에 해당하는 Policy를 삭제한다.

```
policy del <policy-name>
```

policy test

Policy가 어떤 파일들을 배포할 것인지 테스트해 보는 명령어. 배포가 될 파일들의 리스트가 나타난다. 실제로 배포하지는 않는다.

```
policy test <policy-name>
```

Task Command

Task는 Oden의 실제 배포작업을 수행하는 단위이다. Task는 여러개의 Policy들로 구성되어 있다. Task 명령어를 통해 실행할 Policy를 가진 Task들을 설정하고 실행할 수 있다.

task add

새 Task를 생성하기 위한 명령어. 동일한 이름을 가진 Task가 있을 경우 덮어 쓴다.

```
task add <task-name>
  -p[olicy] <policy-name> ...
  -desc <description>
```

-p는 Task에 포함될 Policy들을 지정하는 옵션이다. Policy가 여러개일 경우 스페이스로 구분한다.

두개의 Policy를 가지는 Task 생성

```
task add t0 -p p0 p1
```

Policy p0와 p1을 가지는 Task t0 생성. Policy Command를 통해 p0와 p1이 이미 등록되어 있어야 한다.

task info

Task의 정보를 조회하기 위한 명령어. task-name을 지정하지 않았을 경우 등록된 모든 Task의 정보를 보여준다.

```
task info [<task-name>]
```

task del

등록된 Task 삭제 명령어. task-name에 해당하는 Task를 삭제한다.

```
task del <task-name>
```

task test

Task가 어떤 파일들을 배포할 것인지 미리 확인해 보는 명령어. 실제로 파일을 배포하지는 않는다.

```
task test <task-name>
```

task run

Task를 실행하는 명령어. task-name에 해당하는 Task를 실행한다.

```
task run <task-name>
```

History Command

History 명령어를 통해 배포된 내역을 조회할 수 있다. Task 명령어를 통해 배포된 내역을 조회할 수 있으며, 같은 Task 안에서 수행되었더라도 실제로 배포되지 않은 파일은 기록되지 않는다.

history show

Oden을 통해 배포된 내역을 조회한다. 옵션이 없을 경우 최근에 배포된 Policy의 내역을 보여준다. Oden Shell이나 Telnet을 통해 수행된 Task의 유저 IP 정보는 항상 Oden Server의 IP가 된다. 누가 배포하였는지 정확한 정보를 기록하기 위해서는 Eclipse Plugin과 같이 Oden의 HTTP 서비스를 이용하는 UI를 통해 배포하여야 한다.

```
history show
  [-u[ser] <user-access-ip>]
  [-a[gent] <host-name>]
  [-p[ath] <path>]
  [-d[ate] <start-date: yyyyMMdd> <end-date: yyyyMMdd>]
```

‘-u’ 옵션은 배포 명령어를 수행시킨 사용자의 IP를 검색하기 위한 옵션이다. 70.7로 시작하는 모든 사용자가 배포한 내역을 검색하려면 ‘-u 70.7’ 로 인자를 주면 된다. Telnet으로 수행하였더라도 Oden Shell을 통해 수행된 Task는 모두 Oden Server의 IP로 기록되어 있으니 정확한 사용자를 파악하기 위해서는 Eclipse Plug-in과 같이 Oden의 HTTP 서비스를 사용하는 UI를 이용하여야 한다.

‘-a’ 옵션은 Agent의 IP를 지칭한다. Oden Server에 내장된 Agent인 경우 IP에 localhost로 기록이 된다.

‘-p’는 배포한 파일의 경로를 지정하는 옵션이며 지정한 경로가 들어가 있는 모든 내역을 출력한다.

예를 들어 ‘-p anyframe’을 입력하면 경로에 anyframe이 들어가 있는 모든 내역을 출력한다.

‘-d’는 배포가 실행된 날짜를 지정해 주는 옵션이다. 인자가 하나만 있을 경우 ‘start-date’를 의미하며 ‘start-date’부터 오늘까지 배포된 모든 내역을 출력한다.

특정날짜로 부터 특정 Agent에 배포된 모든 내역을 조회

```
history show -a 192.168.0.2:9862 -d 20090101
```

2009년 1월 1일 부터 192.168.0.2:9862에 배포된 모든 내역을 조회한다.

Oden Server에 내장된 Agent에 배포된 내역을 조회

```
history show -a localhost -d 20090101 20090102
```

2009년 1월 1일부터 2009년 1월 2일 사이에 OdenServer에 내장된 Agent에 배포된 파일 내역을 조회한다.

Note

Oden Server에 내장된 Agent에 배포를 하면 history에는 localhost로 기록이 된다.

특정 IP대역의 사용자가 배포한 내역을 조회

```
history show -u 172.16.70 -d 20090101
```

172.16.70으로 시작하는 IP를 가진 유저들이 2009년 1월 1일 부터 배포한 내역을 조회

특정파일을 배포한 내역을 조회

```
history show -p index.html
```

지금까지 index.html을 배포한 모든 내역을 조회

Snapshot Command

Target Server의 파일을 Backup하거나 Backup된 파일(Snapshot이라 부른다.)을 조회하기 위한 명령어이다. Backup을 수행하기 위해서는 해당 정보를 가지고 있는 Snapshot Plan을 먼저 정의해야 한다. Backup이 수행되면 Plan정보를 바탕으로 폴더를 압축하여 지정한 위치에 저장한다. Snapshot 명령어로 Backup된 파일의 정보를 확인할 수 있다.

Warning

Snapshot Plan이 삭제되거나 수정되면 해당 Plan을 바탕으로 생성된 Snapshot File들은 모두 사라지게 된다.

snapshot add

Snapshot Plan을 추가한다. 기존에 Plan이 있을 경우 해당 정보를 덮어쓰게 된다. 해당 정보를 덮어쓰게 될 경우 Plan을 통해 생성된 Snapshot File들이 모두 사라지므로 주의해야 한다.

```
snapshot add <plan-name>
  -s[ource] / [<location-variable-name>]
  -d[est] <agent-name> [<location-variable-name>]
  [-desc <description>]
```

‘-s’ 옵션은 어느 위치의 파일들을 Backup할 것인지 지정하는 옵션이다. conf/config.xml에 지정된 location명을 입력해야 하며 location명 앞에 ‘/’를 붙여 줘야 한다. ‘/’만 있을 경우 Agent의 default-location을 의미한다.

‘-d’ 옵션에 지정한 Agent를 ‘-s’에서도 사용한다. ‘-d’ 옵션은 Backup되어 압축된 파일을 어디에 저장할 것인지 지정하는 옵션이다. conf/config.xml에 정의된 Agent명과 location명을 지정하여야 한다. Agent와 location이름 사이에는 ‘/’로 구분한다.

간단한 백업

```
snapshot add s0 -s / -d agent0/backup
```

agent0의 default-location 내의 모든 파일들을 압축하여 agent0의 backup 경로에 저장한다. agent0와 backup은 conf/config.xml에 정의되어 있어야 한다.

간단한 백업 2

```
snapshot add s0 -s /anyframe-app -d agent0/backup
```

agent0의 anyframe-app경로 내의 모든 파일들을 압축하여 agent0의 backup 경로에 저장한다. agent0와 anyframe-app, backup은 모두 conf/config.xml에 정의되어 있어야 한다.

snapshot info

Snapshot Plan 혹은 Snapshot File 정보를 조회한다. '-p'나 '-f' 옵션을 붙이지 않았을 경우 모든 Snapshot Plan정보를 출력한다. '-p'를 붙였을 경우 해당 Plan의 정보를 출력하며 plan-name이 없을 경우 모든 Plan정보를 출력한다. '-f' 옵션을 붙였을 경우 Backup 을 통해 생성된 Snapshot File의 정보를 출력하며 file-name을 지정하지 않았을 경우 모든 Snapshot File의 정보를 출력한다.

```
snapshot info
  -p[lan] [<plan-name>] | -f[file] [<file-name>]
```

snapshot del

Snapshot Plan 혹은 Snapshot File을 삭제한다. '-p' 옵션이 있을 경우 지정한 Snapshot Plan을 삭제한다. 이 Plan을 통해 만들어진 Snapshot File도 모두 삭제되므로 주의하여야 한다. '-f' 옵션이 있을 경우 지정한 Snapshot File을 삭제한다.

```
snapshot del
  -p[lan] [<plan-name>] | -f[file] [<file-name>]
```

snapshot test

Snapshot Plan이 수행되었을 경우 어느 경로의 파일들이 어디에 저장될 것인지를 미리 보여준다. 실제로 Backup을 수행하지는 않는다.

```
snapshot test <plan-name>
```

snapshot run

Snapshot Plan의 정보를 바탕으로 Backup을 수행한다.

```
snapshot run <plan-name>
```

Rollback Command

Snapshot 명령어로 Backup된 파일들을 복원할 때 사용되는 명령어이다.

rollback run

Backup으로 생성된 Snapshot File들을 Snapshot Plan을 참조하여 원래의 위치로 복원시킨다. 기존에 있는 파일들을 모두 무시하고 덮어쓰게 되므로 주의하여야 한다.

```
rollback run <snapshot-file-name>
```

Agent Command

conf/config.xml에 설정된 Agent에 관한 설정을 조회하기 위한 명령어이다. conf/config.xml에 설정된 Agent정보는 다른 명령어들을 실행할 때 사용되며, 이 명령어를 통해 그 정보들을 조회해 볼 수 있다. Agent정보는 수정할 수는 없으며, 수정하기 위해서는 conf/config.xml을 직접 수정하여야 한다. conf/config.xml 수정 후 재구동할 필요없이 바로 적용된다.

agent info

```
agent info
```

Viewing Help

Oden Shell에서 제공하는 모든 명령어의 목록을 보기 위해서는 help를 입력하면 된다. 각 명령어의 상세 사용법을 보기 위해서는 명령어 뒤에 help를 붙이거나 뒤의 인자를 제외한 명령어 이름만 입력하면 된다.

Policy Command의 사용법 조회

```
policy 혹은 policy help
```

Policy Command의 사용법을 출력한다.

Working with Eclipse Plug-in

본 장에서는 Oden의 Eclipse plug-in 기반 GUI 환경에 대한 보다 상세한 이해 위한 설명을 제공한다.

Introduction to Oden Eclipse Plug-in

Oden Eclipse Plug-in은 배포관리를 위한 Oden Server와 연계 하여 배포관리를 지원하는 플러그인으로, 다음과 같은 주요 특징점을 통해 배포 용이성 및 높은 신뢰성을 기대하게 해 준다.

- 배포/이관 담당자에게 친숙한 Eclipse Plug-in UI를 제공하여 사용자 이용 용이성을 제공함
- 배포 이력 관리를 통한 배포의 정합성 및 무결성을 확보하여 해당 응용프로그램의 연속성을 보장함
- Update 옵션 및 Snapshot 기능 제공으로 불필요한 리소스 낭비요소를 제거하고 배포에 관한 신뢰성을 보장함
- Build Repository를 트리 구조로 제공하고 즉시 배포가 가능하게 하여 신속한 장애 복구가 가능하게 함

Oden Eclipse Plug-in은 크게 View와 Editor로 구분되며 Oden Explorer View, Oden Policy and Task Editor, Oden Snapshot View , Oden Deployment History View 로 구성된다.

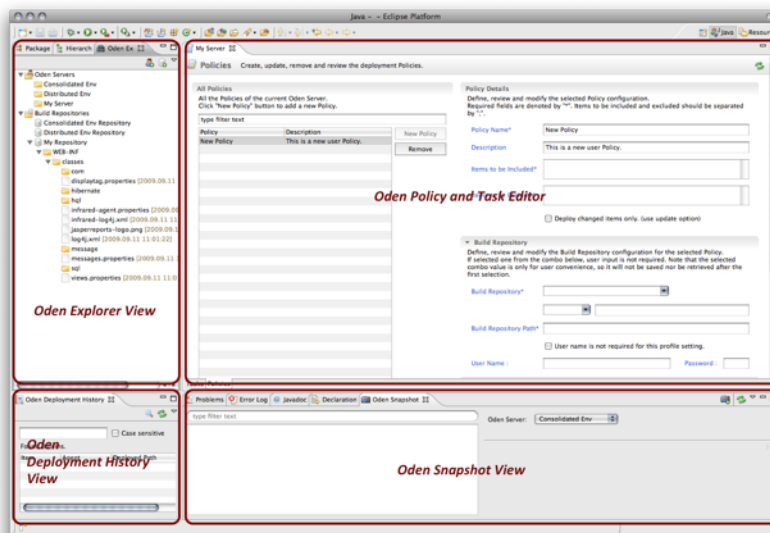


Figure 8. Oden Eclipse Plug-in

Oden Explorer View

Oden Explorer View는 배포 환경 구성을 위한 Server와 Repository를 구성 할 수 있고 Task와 Policy을 편집 할 수 있는 Editor 창으로 접근이 가능하다.

- Oden Explorer View 열기

Eclipse 메뉴를 통해 Window > Show View > Other... > Anyframe > Oden Explorer

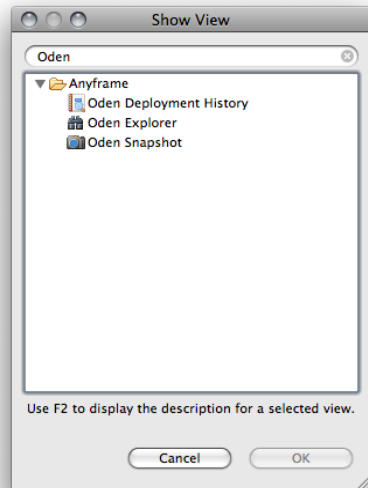


Figure 9. Open the Oden Explorer View (and others are also the same)

- Oden Explorer View 구성

Oden Explorer View는 크게 Oden Servers와 Build Repositories로 구성 되어 있으며 Oden Server 및 Build Repository 등에 대한 연결 프로파일을 대화창을 통해 관리할 수 있다. 관리할 수 있는 기능은 각 프로파일에 대한 신규 생성, 편집, 복제, 삭제 등이다.

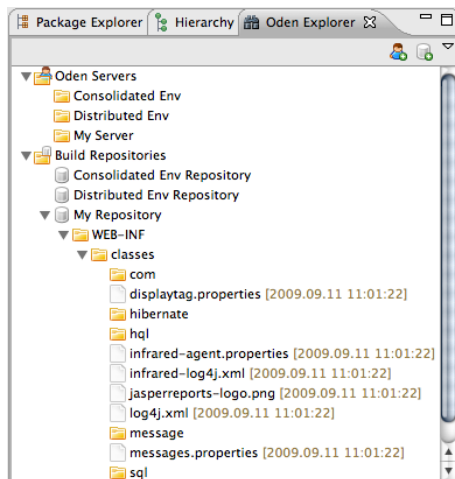


Figure 10. Oden Explorer View

Servers

Oden을 활용한 배포환경 구성 시 설치한 Oden Server의 프로파일을 관리한다. 하나의 Oden Server에 대해서도 관리의 편의를 위해 여러개의 프로파일을 만들 수 있다.

Managing Oden Server Profiles

Context Menu를 통해 Oden Server의 프로파일을 관리하며 메뉴는 다음과 같다.

- **New Oden Server Profile...**
"Oden Servers" 항목에서 호출한 Context Menu를 통해, Oden Server 프로파일을 새로 만들 수 있음
- **Edit Oden Server Profile...**
생성한 Oden Server 프로파일에 대한 편집 기능을 제공함
- **Duplicate Oden Server Profile...**
생성한 Oden Server 프로파일에 대한 복제 기능을 제공함. 기본적으로 "[기존 Oden Server 프로파일 이름] - duplicated"의 이름이 부여되며, 그 외 정보는 기존 정보를 바탕으로 기본값으로 설정됨
- **Delete Oden Server Profile...**
생성한 Oden Server 프로파일에 대한 삭제 기능을 제공함. 해당 프로파일을 통해 "Oden Policy and Task Editor"가 열려 있을 경우, 해당 Editor를 닫음

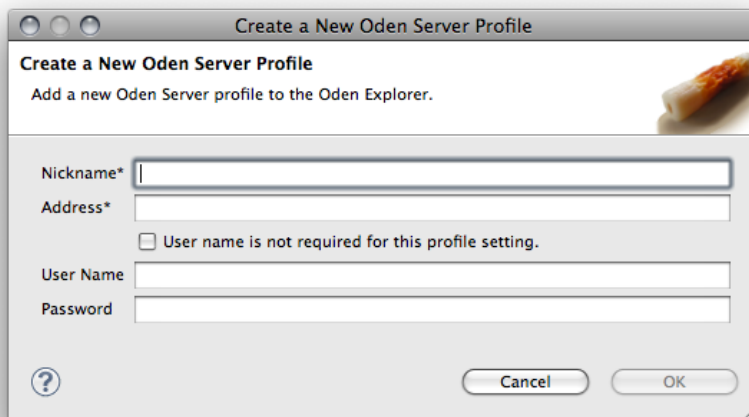


Figure 11. Create a New Oden Server Profile

Server를 생성하기 위한 정보들은 아래와 같다.

- **Nickname** : 생성하는 Server의 별칭을 입력한다.
- **Address** : Oden Server가 설치된 곳의 hostname 또는 IP 주소를 포트 번호를 포함하여 입력한다.
eg. localhost:9860
- **User name is not required for this profile setting.** : Oden Server에 접근 시 인증 과정이 필요 없을 경우 체크한다.
- **User Name** : 계정이 필요한 경우 아이디를 입력한다.

- Password : 계정이 필요한 경우 암호를 입력한다.

Note

Oden Server 프로파일에 대한 변경이 있을 경우, Oden Explorer View가 갱신되면서 기존에 열려있던 Build Repository의 하위 폴더들이 닫히게 된다.

Build Repositories

Build Repository는 CTIP 환경에서 CI서버(빌드서버)에서 Build 결과물이 생성되는 위치를 지칭하며 이에 대한 프로파일은 업무별 또는 용도에 따라 여럿을 설정 할 수 있다. Eclipse Plug-in에서는 Build Repository 프로파일을 설정하여 사용자가 직관적으로 배포가 가능하도록 한다.

Managing Build Repository Profiles

Context Menu를 통해 Build Repository의 프로파일을 관리하며 메뉴는 다음과 같다.

- **New Build Repository Profile...**
"Build Repositories" 항목에서 호출한 Context Menu를 통해, Build Repository 프로파일을 새로 만들 수 있음
- **Edit Build Repository Profile...**
생성한 Build Repository 프로파일에 대한 편집 기능을 제공함
- **Duplicate Build Repository Profile...**
생성한 Build Repository 프로파일에 대한 복제 기능을 제공함. 기본적으로 "[기존 Build Repository 프로파일 이름- duplicated]"의 이름이 부여되며, 그 외 정보는 기존 정보를 바탕으로 기본값으로 설정됨
- **Delete Build Repository Profile...**
생성한 Build Repository 프로파일에 대한 삭제 기능을 제공함

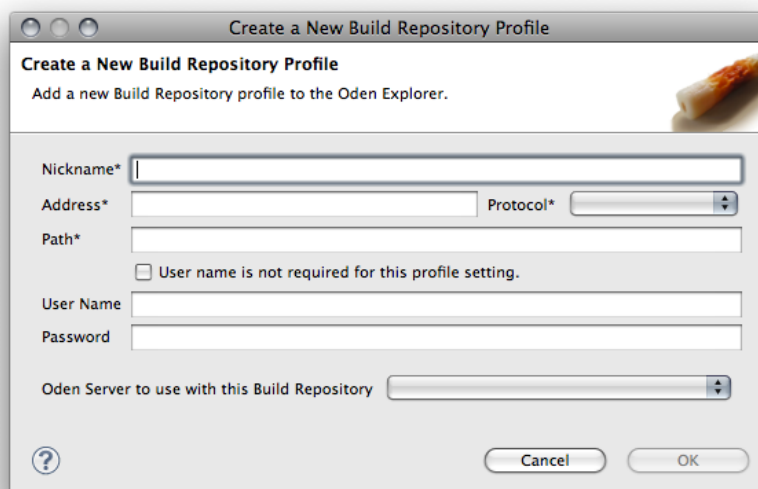


Figure 12. Create a New Build Repository Profile

Build Repository를 생성하기 위한 정보들은 아래와 같다.

- Nickname : 생성하는 Build Repository의 별칭을 입력한다.
- Address : Host name 또는 IP 주소를 입력한다. eg. localhost,127.0.0.1
- Protocol : Build Repository로 접근하기 위한 Protocol을 선택한다.
- Path : Build Repository의 root path를 입력한다.
- User name is not required for this profile setting : Sever에 접근 시 계정이 필요 없을 경우 체크한다.
- User Name : 계정이 필요한 경우 아이디를 입력한다.
- Password : 계정이 필요한 경우 암호를 입력한다.
- Oden Server to use with this Build Repository : Build Repository를 사용하는 Oden Server를 선택한다.

Note

Build Repository 프로파일에 대한 변경이 있을 경우, Oden Explorer View가 갱신되면서 기존에 열려있던 Build Repository의 하위 폴더들이 닫히게 된다.

Deploy Now Action

Deploy Now는 Build Repository 트리 구조에서 사용자가 원하는 폴더 및 파일을 선택하고 마우스 우 클릭을 통한 context menu를 이용해 쉽게 배포를 수행 하는 기능이다.

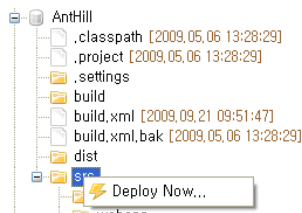


Figure 13. Deploy Now Action

Deploy Now...를 선택하면 배포 될 파일 목록을 미리보기를 통해 확인 할 수 있다.

Warning

Deploy Now 기능은 update 기능을 제공하지 않아 모든 Item들을 배포 하므로 기능 수행에 관하여 주의가 필요하다.

배포 목록을 확인하고 OK 버튼을 클릭하면 Agent로 배포를 수행하고 보여주는 정보는 다음과 같다.

- Repository : 선택한 Build Repository의 root path
- Path : 선택한 Build Repository의 root path 이후의 Item path
- Item : 배포되는 파일의 이름
- Agent : 배포 되는 config.xml 상의 Agent 별칭

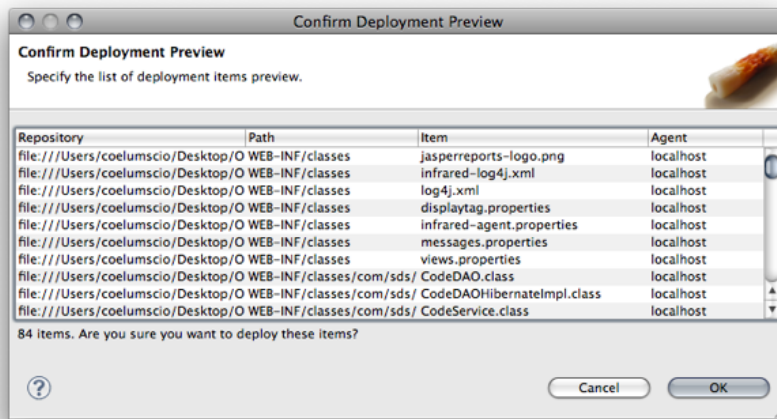


Figure 14. Confirm Deployment Preview

Oden Policy and Task Editor

Policy(배포정책)는 배포를 수행 할 수 있는 단위를 말하며, 배포서버에 관한 정보, 배포 시 포함 할 경로 및 파일, 제외 할 경로 및 파일, 배포서버에 관한 정보를 포함한다. 또한 Task(배포작업)는 정의된 Policy들의 묶음 이며 실질적으로 배포를 수행 하는 구조체 이다. Oden Policy and Task Editor는 Policy 및 Task에 대해 조회, 생성, 삭제, 편집등을 수행하는 Editor 창이다. Editor에 접근 하기 위해서는 원하는 Oden Server 프로파일의 Context Menu에서 "Open Oden Policy and Task Editor..."를 선택하여 Editor를 연다.

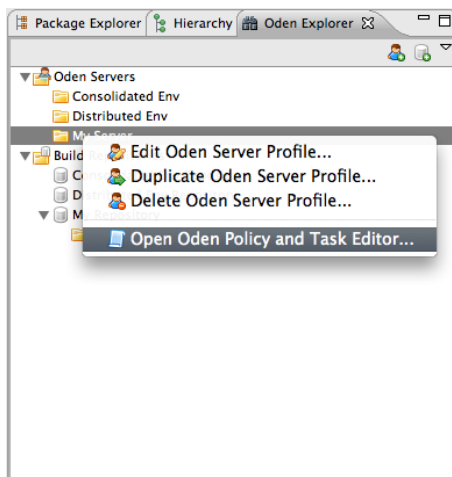


Figure 15. Open Oden Policy and Task Editor

Policies Tab

Policies Tab은 All Policies, Policy Details, Build Repository, Agents 등으로 구성되며 상세 기능 설명은 다음과 같다.

All Policies

Server에 등록되어 있는 Policy 목록을 보여주며 filter text를 통해 Policy 조회를 수행 할 수 있고 New Policy 버튼 클릭을 통해 신규 Policy를 입력하며, Remove를 통해 생성되어 있는 Policy를 제거한다.필요시 오른쪽 상단의 refresh 버튼을 클릭하여 최신 Policy 목록을 조회한다.

Policy Details

Policy에 관한 일반적인 사항을 입력, 편집, 조회 하는 창이며 필드의 상세내용은 다음과 같다.

필드	설명	비고
Policy Name	입력하는 Policy의 이름	중복 Policy Name 입력 불가
Description	입력하는 Policy에 대한 설명	
Items to be Included	Build Repository Path 이후에 포함할 아이템 (경로 및 파일)	입력 필드의 구분은 “;”로 구분 *: 1레벨 하위 모든 것 ** : 하위 레벨 모든 것 eg. Webapp/**;webapp/WEB-INF/lib/*
Items to be Excluded	Build Repository Path 이후에 제외할 아이템 (경로 및 파일)	“Items to be Included” 필드와 입력형태 동일함
Deploy changed items only.	변경 파일만 배포를 원할 경우 체크	Update 옵션

Table 1. Policy Details Fields

Figure 16. Policy Detail

Build Repository

Build Repository의 정보를 입력하며 Oden Explorer View에 등록된 Build Repositories를 선택하여 정보를 로딩하여 입력하거나 “User Input” 를 선택하여 직접 입력이 가능하다.

Figure 17. Build Repository

Agents

배포되는 Agent를 등록하며 Add... 버튼을 클릭하고 대화창을 통해 추가한다. 또한 필요시 Agent를 선택하고 Remove 버튼을 클릭하여 제거한다.

Figure 18. Select Agents

conf/config.xml 에 설정되어 있는 Agent 및 Location Variable을 선택하고 OK 버튼을 클릭하여 Agent 테이블로 추가한다. 원하는 Agent, Location Variable을 선택하면 하단에 배포 서버의 URL 및 배포경로를 제공한다.

Adding a New Policy

New Policy를 클릭하여 신규 Policy를 등록한다. 신규 Policy 입력 시에 임시 Policy Name과 Description이 자동 입력되며 원하는 값으로 변경하여 입력하면 된다.

Warning

임시 Policy Name은 사용자가 원하는 이름으로 반드시 변경하여야 한다.

Editing an Existing Policy

All Policies 테이블에서 원하는 Policy를 클릭 후, Policy Details, Build Repository, Agents 의 값들이 조회가 되며 원하는 값으로 편집을 하고 Save this Policy를 클릭한다.

Deleting a Policy

All Policies 테이블에서 원하는 Policy를 선택 후, Remove를 클릭한다. 확인창을 통해 다시 한번 고려할 수 있으며, 삭제된 Policy는 복구가 불가능하다.

Tasks Tab

All Tasks, Task Details, Policies 등으로 구성되며, 상세 기능 설명은 다음과 같다.

- **All Tasks**
Sever에 등록되어 있는 Task 목록을 보여주며 filter text를 통해 Task Name 조회를 수행 할 수 있고 New Task 버튼 클릭을 통해 신규 Task를 입력하며, Remove를 통해 생성되어 있는 Task를 제거한다. 필요 시 오른쪽 상단의 refresh 버튼을 클릭하여 최신 Task 목록을 조회한다.
- **Task Details**
Task에 관한 일반적인 사항을 입력, 편집, 조회 할 수 있다. TaskName은 입력하는 Task의 이름, Description은 입력하는 Task의 개요를 나타낸다.
- **Policies**
Task에 포함될 Policy를 체크를 통해 포함 시킨다. 필요 시 오른쪽 상단의 refresh 버튼을 클릭하여 최신 Policy 목록을 조회한다. 입력 혹은 편집 정보는 Save this Task 버튼을 클릭하여야 저장된다.

Adding a New Task

New Task를 클릭하여 신규 Task를 등록한다. 신규 Task 입력 시에 임시 Task Name 과 Description이 자동 입력되며 원하는 값으로 변경하여 입력하면 된다.

Warning

임시 Task Name은 사용자가 원하는 이름으로 반드시 변경하여야 한다.

Editing an Existing Task

All Tasks에서 원하는 Task를 클릭 후, Task Details, Policies 등의 값들을 편집하고 Save this Task를 클릭한다.

Deleting a Task

All Tasks에서 원하는 Task를 선택 후 Remove를 클릭한다. 확인창을 통해 다시 한 번 고려할 수 있으며, 삭제된 Task는 복구가 불가능하다.

Running a Task

All Tasks에서 배포를 수행하고자 하는 Task를 선택 후 Run this Task를 클릭하면 배포 될 파일 목록을 미리보기를 통해 확인 할 수 있고 OK 버튼을 클릭하면 Agent로 배포를 수행한다.

Oden Snapshot View

Snapshot View에서는 Snapshot Plan과 Snapshot을 관리할 수 있다. Plan은 Snapshot을 생성하고자 하는 디렉토리의 정보를 담고 있으며, 또한 Snapshot이 저장될 위치 역시 포함하고 있다.

Snapshot View에 관한 설명은 다음과 같다.

- **Oden Server**
Oden Server를 선택할 수 있다. View를 처음 시작하면 목록의 첫번째 Oden Server를 선택한 상태가 된다.
- **Snapshot Plan and Snapshot Tree**
Oden Server를 선택했을 경우, 그에 해당하는 Snapshot Plan과 Snapshot으로 이루어진 tree가 나타난다.
- **Details**
Tree 중에서 Snapshot Plan이나 Snapshot을 선택했을 경우, 각각의 상세정보가 나타난다.

만약 Oden Server가 연결될 수 없다면, Error Log를 통해 Oden Server에 연결할 수 없다는 메시지가 나타난다.

Toolbar에 있는 Refresh 버튼을 통해 Oden Server의 목록과 선택된 Oden Server의 Snapshot Plan, Snapshot 구조가 Refresh되어 update된 내용을 얻을 수 있다.

Managing Snapshot Plans

Snapshot Plan은 Snapshot 대상 디렉토리, Agent, 저장 디렉토리의 정보를 담고 있다. Snapshot은 Snapshot Plan을 통해서만 생성이 가능하다.

Adding a New Snapshot Plan

Snapshot Plan은 두가지 방법으로 생성이 가능하다.

- Toolbar에 있는 버튼을 통해 Snapshot Add Dialog를 열 수 있다.
- Tree 위치에서 Context Menu를 통해서 Snapshot Add Dialog를 열 수 있다.

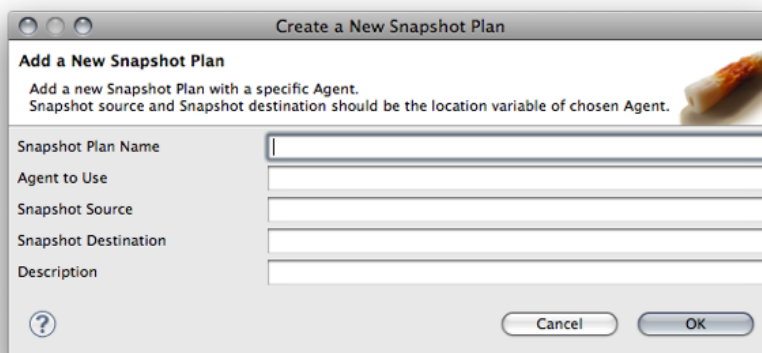


Figure 19. Add a New Snapshot Plan

만약, Oden Server가 연결되어 있지 않은 상태에서 Snapshot Plan을 Add 하려고 하면, Error Log에 Oden Server에 연결할 수 없다는 메시지가 나타나게 된다.

위 다이얼로그를 통해 새로운 Plan을 생성할 수도 있고, 기존에 있던 Plan을 update할 수도 있다.

- **Snapshot Plan Name**
Plan을 식별할 수 있는 이름
- **Agent to Use**
Snapshot을 만들 대상 디렉토리, 저장할 디렉토리가 저장된 변수들을 가지고 있는 Agent
- **Snapshot Source**
Snapshot을 만들 대상 디렉토리가 저장된 변수명
- **Snapshot Destination**
생성된 Snapshot을 저장할 디렉토리가 저장된 변수명
- **Description**
기타 정보

정상적으로 Snapshot Plan이 저장되면, View에는 다음과 같이 나타나게 된다. Plan을 선택했을 경우, 상세 정보가 나타나게 되는데, 여기에 Snapshot Plan에 관한 정보가 추가되어 보여진다.

- **User Information**
Snapshot Plan을 생성한 위치의 IP가 나타난다.
- **Date**
Snapshot Plan을 생성한 날짜, 시간이 YYYY/MM/DD hh:mm:ss 형태로 나타난다.

Editing an Existing Snapshot Plan

상세 정보에서 Source location, Destination Agent, Destination, Description은 추후에 수정이 가능하다. 수정 후, Save 버튼을 클릭하게 되면, 정보가 새로 저장된다. 이때, 만약 Plan 하위의 Snapshot이 존재하면, 다음과 같이 하위의 Snapshot이 다 지워지더라도 수정할 것인가를 확인하는 메시지가 나타난다.

Warning

Snapshot Plan의 내용을 수정하게 되면, 해당 Plan을 바탕으로 생성된 Snapshot이 모두 삭제된다.

Duplicating an Existing Snapshot Plan

기존에 생성된 Snapshot Plan과 동일한 환경을 가지는 Plan을 생성하기 위해서는 Add 버튼을 통해 새로 생성하는 방법과, duplicate 메뉴를 사용하는 방법이 있다. 마우스 우클릭을 통해 Duplicate Snapshot Plan 메뉴를 클릭하게 되면, Snapshot Duplicate Dialog가 나타난다.

Snapshot Name은 기존의 Plan명에 -duplicate만 추가되었으며, 나머지 정보들은 동일하다. 그대로 사용해도 되고, 수정해서 사용해도 된다.

Deleting a Snapshot Plan

생성한 Snapshot Plan을 삭제하기 위해서 Delete 버튼을 클릭하게 되면, 선택한 Snapshot Plan에 대해서 삭제 확인 메시지가 나타나게 된다.

Warning

Snapshot Plan의 내용을 삭제하게 되면, 해당 Plan을 바탕으로 생성된 Snapshot이 모두 삭제된다.

Managing Snapshots

Snapshot은 Snapshot Plan의 정보를 토대로 Backup을 수행한 결과로 나타나는 File이다.

Taking a Snapshot

Snapshot Plan을 토대로 Snapshot을 생성하기 위해서는 Take Snapshot 버튼을 클릭한다.

Snapshot을 생성하기 위해 선택한 Snapshot Plan이 맞는지 확인 후, Snapshot을 생성하게 된다.

Snapshot은 Source 디렉토리 하위의 모든 폴더, 파일등을 .zip으로 묶은 후, 확장자 없이 Destination에 저장된다. Snapshot을 생성하고 난 뒤에는 Tree 위치에서 Snapshot을 확인할 수 있다.

Snapshot의 상세 정보는 다음과 같다.

- **Snapshot File Name**
Snapshot의 이름으로 '생성일_번호'의 형태로 저장된다.
- **Size**
Snapshot의 size로 KB단위로 나타난다
- **Date**
Snapshot을 생성한 날짜, 시간이 YYYY/MM/DD hh:mm:ss 형태로 나타난다.

Deleting a Snapshot

생성한 Snapshot을 삭제하기 위해서 Delete 버튼을 클릭하게 되면, 선택한 Snapshot에 대해서 삭제 확인 메시지가 나타나게 된다.

Rollback with Snapshots

생성된 Snapshot을 통해 특정 시점으로 데이터를 돌리고자 할 때, Rollback 기능을 사용할 수 있다.

원하는 Snapshot을 클릭한 후 Rollback 버튼을 클릭하면 확인을 요청하는 창이 나타나고, Rollback을 수행할 수 있다.

Oden Deployment History View

Oden Deployment History View는 사용자들이 원하는 배포 Item을 검색하고 조회 할 수 있는 기능을 제공한다. Oden Deployment History View는 다음과 같은 방법으로 접근한다.

- **Oden Deployment History View 열기**
Eclipse 메뉴를 통해 Window > Show View > Other... > Anyframe > Oden Deployment History

Searching Deployment History

Oden Deployment History View 구성은 다음과 같다.

- **검색창**
검색창에 검색을 원하는 Item의 이름을 입력한 후 엔터를 누르면 해당하는 배포 목록이 검색되며, 빈칸인 채로 검색을 실시하면 최신 배포 목록이 검색된다. Oden Deployment History View의 검색 범위는 모든 사용자의 배포이력이다.
- **Server 선택 콤보 박스**
사용자가 원하는 서버를 선택 할 수 있으며 Oden Explorer View 에서 생성한 Oden Servers의 별칭이 자동으로 구성되어 선택 할 수 있다
- **Refresh 버튼**
Oden Explorer View 상에서 Servers의 내용의 변경이 있을 경우 최신 정보를 얻어 오기 위해 사용한다.
- **검색 결과창**
검색 결과는 Oden Deployment History View의 하단 검색 결과창에 나타난다.

Advanced Search

고급검색기능은 Advanced Search 버튼을 클릭하여 실행시킬 수 있으며, 다양한 검색조건을 AND 조건으로 조합하여 검색할 수 있다. 고급검색기능을 통하여 조합할 수 있는 검색조건은 다음과 같다.

- **Item Name**
배포 파일 이름에 대하여 해당 문자열을 기준으로 검색한다.
- **IP**
배포를 수행한 IP에 대하여 검색한다.
- **Deployed Date**
배포를 수행한 시점에 대하여 이전, 이후 등의 기준으로 검색한다.

Indexes

A

Agent · 7, *See* Oden Explorer View

C

Command Line Interface · *See* Shell Commands

E

Eclipse Plug-in

Configuration · 16

Installation · 15

Oden Deployment History View · 37

Oden Explorer View · 27

Oden Policy and Task Editor · 31

Oden Snapshot View · 34

G

Graphical User Interface · *See* Eclipse Plug-in

O

Oden

Architecture · 4

Configuration · 13

Installation · 12

Key Features · 3

LICENSE · 2

Starting Up · 15

Technical Support · 2

P

Policy · 7, *See* Shell Commands, Oden Policy and Task Editor

S

Server · 6, *See* Oden Explorer View

Shell Commands

History Command · 22

Policy Command · 18

Rollback Command · 25

Snapshot Command · 23

Task Command · 21

Snapshot · *See* Shell Commands, Oden Snapshot View

Rollback · 9

Snapshot · 9

Snapshot Plan · 8

System Requirements · 11

T

Task · 8, *See* Shell Commands, Oden Policy and Task Editor