

Anyframe Gen



Version 1.5.1

저작권 © 2007-2010 삼성SDS

본 문서의 저작권은 삼성SDS에 있으며 Anyframe 오픈소스 커뮤니티 활동의 목적하에서 자유로운 이용이 가능합니다. 본 문서를 복제, 배포할 경우에는 저작권자를 명시하여 주시기 바라며 본 문서를 변경하실 경우에는 원문과 변경된 내용을 표시하여 주시기 바랍니다. 원문과 변경된 문서에 대한 상업적 용도의 활용은 허용되지 않습니다. 본 문서에 오류가 있다고 판단될 경우 이슈로 등록해 주시면 적절한 조치를 취하도록 하겠습니다.

I. Overview	1
1. Introduction	2
1.1. 개요	2
1.2. 기술 지원	3
1.3. 유지 보수	3
1.4. 시스템 사양	3
2. Key Features	5
2.1. 주요 특징	5
2.2. Anyframe Gen의 구조	5
2.3. 프로젝트 중심의 개발	6
2.4. CRUD 코드 생성	6
2.5. 테스트 코드 생성	7
II. Installation	8
3. 설치	9
III. Command Line Interface	11
4. 환경 설정	12
4.1. Anyframe Gen 환경 변수 설정	12
4.2. Anyframe Gen 도움말	12
4.3. DB 구동	13
5. Commands for Project	14
5.1. 프로젝트 생성	14
5.2. Anyframe 플러그인 설치 및 제거	16
5.3. Anyframe 플러그인 목록 조회 및 업데이트	19
5.4. Domain Class 생성	20
5.5. CRUD 소스 코드 생성	22
5.5.1. Business Layer 코드 생성	22
5.5.2. Presentation Layer 코드 생성	25
5.5.3. Business Layer와 Presentation Layer 코드 함께 생성	27
5.6. DB 정보 변경	28
6. 어플리케이션 및 프로젝트 빌드	32
6.1. 어플리케이션 빌드	32
6.1.1. 예시1) 어플리케이션 빌드 후 클래스 배포	32
6.1.2. 예시2) 어플리케이션 빌드 후 JAR 파일 배포	33
6.1.3. 예시3) 어플리케이션 빌드 후 WAR 파일 배포	33
6.2. 프로젝트 빌드	33
6.2.1. Ant 프로젝트 빌드	33
6.2.2. Maven 프로젝트 빌드	34
7. 어플리케이션 빌드 및 실행	36
7.1. 어플리케이션 빌드 및 웹 어플리케이션 실행	36
7.1.1. Ant 프로젝트 빌드 및 실행	36
7.1.2. Maven 프로젝트 실행	37
IV. Eclipse Plugin	38
8. Installing Anyframe Gen Plugin	39
8.1. Anyframe Gen 플러그인 설치	39
8.2. Hudson 연계 플러그인 설정하기	40
9. Preferences	42
9.1. Eclipse 설정	42
9.2. DB 구동	42
9.3. MAVEN_HOME 설정	43
10. New Project Creation	44
10.1. Create new project	44
10.2. 프로젝트 생성 시 발생할 수 있는 에러 해결 Tip	47
10.2.1. Out of memory	47
10.2.2. Project Clean & JSP Validation Error	48
11. Domain Generation	49

12. CRUD Generation	51
13. Configuration	56
14. JDBC Setting	57
15. Anyframe Plugin Installation	58
16. CTIP	61
17. Project Build	64
17.1. Ant 프로젝트 빌드	64
17.2. Maven 프로젝트 빌드	65
V. Templates Extensions	66
18. 프로젝트 템플릿 확장	67
18.1. Web 타입 프로젝트 템플릿 확장	67
18.2. Service 타입 프로젝트 템플릿 확장	68
19. 소스 코드 템플릿 확장	71
19.1. 템플릿 폴더 생성	71
19.2. 템플릿 파일 목록 작성(template.config)	71
19.3. 템플릿 파일(*.ftl) 수정	73
19.4. 템플릿 파일(*.ftl) 변수	75
19.5. 변경된 템플릿 파일(*.ftl) 적용	76

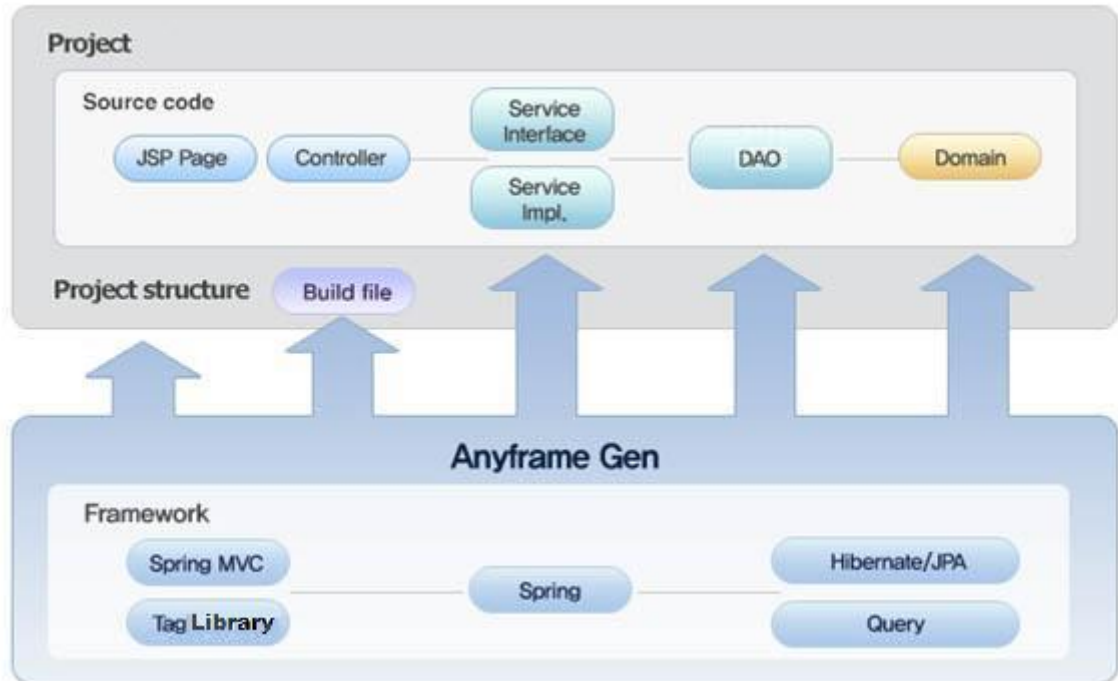
I. Overview

Anyframe Gen은 Anyframe 기반의 프로젝트 개발 시 기존 개발 방식보다 훨씬 쉽고 빠르게 개발을 시작할 수 있도록 Scaffolding, 코드 생성 기능 등을 제공하는 개발 툴이다.

개요에서는 툴을 통해 제공되는 기능, 특징, 자동 생성된 코드를 통해 재사용될 수 있는 공통 기능 등에 대해 설명하고 있다. 이밖에 기술 지원 방법 및 유지 보수 그리고 시스템 사양 등에 대한 사항도 제시하고 있다. 또한 Anyframe Gen의 주요 특징, 구조, 프로젝트 중심 개발에 대해 소개하고 있다.

1.Introduction

Anyframe Gen은 최신 오픈소스(Anyframe)를 바탕으로 기존 방식보다 훨씬 쉽고 빠르게 개발을 시작할 수 있도록 Scaffolding, 코드 생성 기능 등을 제공한다.



1.1.개요

- Anyframe Gen 설치를 통해 기존의 복잡한 설치 및 설정 작업을 최소화한다. Anyframe Gen의 설치로 Anyframe 설치, 프로젝트 구조 생성, 단위 모듈 생성, 소스코드 생성, 빌드/테스트/패키지 스크립트 생성 등이 가능하다.
- 프로젝트를 빌드도구(Ant/Maven)에 따라 선택하여 생성할 수 있어서 Ant 또는 Maven으로 프로젝트 빌드가 가능하다.
- Web 타입, Service 타입으로 프로젝트 타입을 선택할 수 있어 프로젝트 중심의 개발이 가능하다.
- 엔티티 클래스(JPA)와 기본 템플릿(FreeMarker)을 이용한 도메인 객체 중심의 CRUD 기본 코드가 생성되고, 자동으로 단위/통합 테스트 케이스가 작성된다. 템플릿은 프로젝트의 개발표준에 따라 변경이 가능하다.
- 템플릿을 이용해 생성된 CRUD 소스는 default 템플릿의 경우 AbstractDAO 를 기반으로 작성된 코드이며(상속 혹은 참조), miplatform 템플릿의 경우 Anyframe MiPlatform 쿼리서비스(MiPQueryService) 를 기반으로 작성된 코드이다.
- 코드 생성 시, 다음과 같은 공통기능이 제공된다.

표 1.1. generated common functions

function
Service/DAO
Search (Primary Keys, Required Fields)
Paging (Paging Navigation Bar)
Exception / Message Handling
User Defined Template

- DB 테이블에 해당하는 도메인 클래스를 생성하고, 해당 도메인 클래스를 기반으로 CRUD 소스 코드를 생성하는 데 이때 DB 테이블에 맞는 테스트 데이터까지 함께 생성해주고 있다. DB는 기본적으로 HSQL DB를 샘플 DB로 제공하고 있으나 Oracle, MySQL, Sybase DB 등 DB 정보를 변경하여, 프로젝트에서 사용하는 DB 기반 코드 생성 기능을 사용할 수 있게 하고 있다. 이때, 각 DB 별 제공되는 테이블 컬럼 타입이 다양하므로 도메인 클래스 생성 시 혹은 CRUD를 위한 테스트 데이터 생성 시 지원되지 않는 타입 정보로 인한 오류가 생길 수 있다. 이 때에는 DB 컬럼 타입 별 매핑되는 자바 타입 정보를 설정해주는 파일 내용을 변경하여 지원해주도록 한다.

1.2. 기술 지원

Anyframe Gen에 대한 기술 지원은 Anyframe 오픈소스 커뮤니티 사이트(<http://www.anyframejava.org/>)의 포럼 [<http://www.anyframejava.org/forum>] 메뉴를 통해 이루어지며, 단순 질의 응답에서부터 소스 코드에 대한 구체적인 가이드 및 해결책을 제시한다. 특정 프로젝트를 위한 기술 컨설팅 지원이 필요한 경우 Anyframe 오픈소스 커뮤니티 사이트의 연락처 [<http://www.anyframejava.org/about/contactus>]를 통해 요청할 수 있다. 또한, 이슈관리시스템인 JIRA(<http://dev.anyframejava.org/jira/>)를 통해, Bug Fix, Improvements, New Features 에 대한 이슈들을 요청할 수 있다. 자세한 사용방법은 이곳 [<http://www.anyframejava.org/development/issue>] 을 참조하도록 한다.

1.3. 유지 보수

Anyframe 오픈소스 커뮤니티 사이트(<http://www.anyframejava.org/>)를 통해 릴리즈된 최신버전 및 이전 버전에 해당하는 라이브러리, 매뉴얼 등을 제공받을 수 있으며 패치 및 업그레이드되는 실시간 파일의 경우 Reuse Repository [<http://dev.anyframejava.org/artifactory/>] 를 통해 제공된다. 자세한 사용방법은 이곳 [<http://www.anyframejava.org/development/ctip#repository>] 을 참조하도록 한다.

1.4. 시스템 사양

하드웨어와 소프트웨어 요구 사항은 다음 테이블에 작성되어 있다. 아래 작성된 하드웨어 사양 이하의 시스템에서도 Anyframe Gen을 사용할 수 있으나, 빠른 성능을 위해서는 아래 사양을 권고한다. 또한 아래 작성된 Storage의 경우, Anyframe Gen 전체 설치 파일에서부터 Eclipse, Tomcat, MiPlatform 등 모든 관련 파일 설치를 합하여 작성된 것으로 Anyframe Gen 만으로는 1GB 이하의 용량을 차지하고 있다. 소프트웨어 요구 사항은 반드시 충족시켜줘야 한다.

표 1.2. System Requirements(HW)

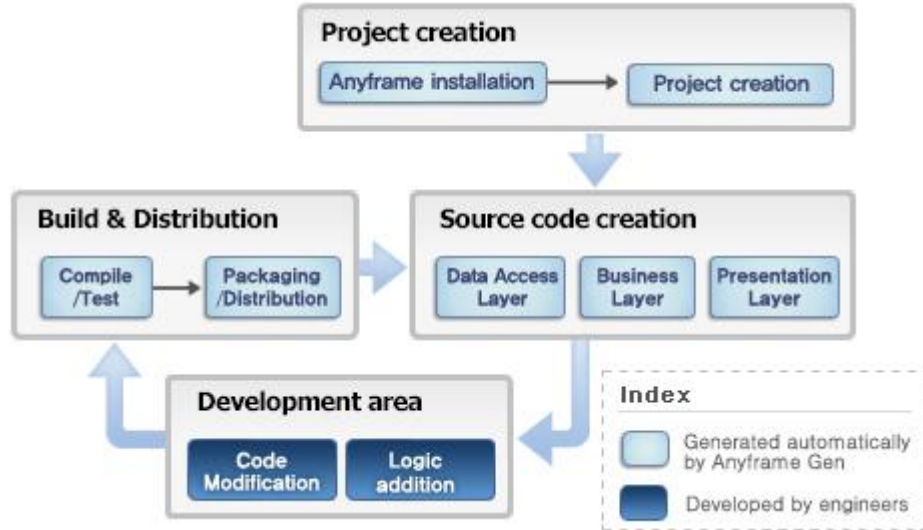
Hardware Requirements	Description
Memory	1024MB RAM 이상
Storage	1GB 하드 디스크 공간

표 1.3. System Requirements(SW)

Software Requirements	Description
JDK	1.5.x Version
Eclipse IDE	Eclipse 3.3.x Version 이상 지원(Europa/Ganymede/Galileo)

2.Key Features

Anyframe Gen은 어플리케이션 프레임워크인 Anyframe 기반으로 프로젝트를 개발할 때 Anyframe 설치, 프로젝트 생성 및 코드 생성 기능을 통해 개발 편의성을 제공해주는 개발 툴로 CLI(Command Line Interface) 혹은 Eclipse Plugin 툴을 통해 Scaffolding과 코드 생성 기능을 제공한다. 아래 그림에서와 같이 Anyframe 설치에서부터 프로젝트 구조, 코드 생성, 빌드 수행에 이르기까지 개발 프로세스 전반에 걸쳐 개발에 도움을 준다.



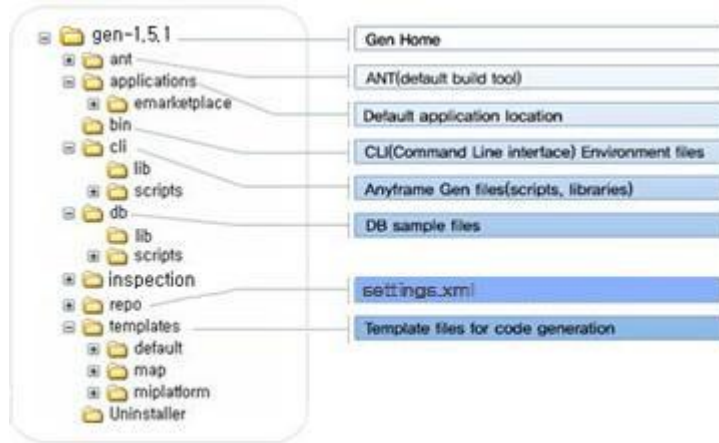
2.1.주요 특징

Anyframe Gen은 다음과 같은 주요 특징을 가지고 있다.

- 개발 대상 프로젝트에 대한 프로젝트 구조 및 빌드 스크립트 생성한다.
- 도메인 클래스를 중심으로 CRUD 기본 코드 생성하고 자동으로 테스트 코드와 테스트 데이터를 생성한다.
- 프로젝트 개발 시 프로젝트 구조, 소스 코드 생성 기능 및 프로젝트 빌드 기능을 제공함으로써 신속한 개발 환경 구성을 가능하게 한다.
- Anyframe을 사용하여 개발 시 직면하게 되는 복잡한 설정 파일로 인한 잦은 오류 발생 상황을 줄일 수 있다.
- 개발 초기 공통 개발 템플릿을 따로 구성하여 배포하고 각 개발자들은 배포된 개발 템플릿 소스를 참조하여 수정해서 사용해야 하는 이러한 반복적인 작업을, Anyframe Gen에서 제공하는 코드 생성 기능을 이용하면 하지 않게 된다.
- Best Practice에 해당하는 샘플 소스 코드를 자동 생성시켜줌으로써 Anyframe을 활용한 개발 시 참조하기 용이하다.
- 단위 테스트 케이스 및 통합 테스트 케이스를 자동 생성시켜줌으로써 개발 진행 시 테스트를 통한 코드 품질을 향상시킬 수 있다.

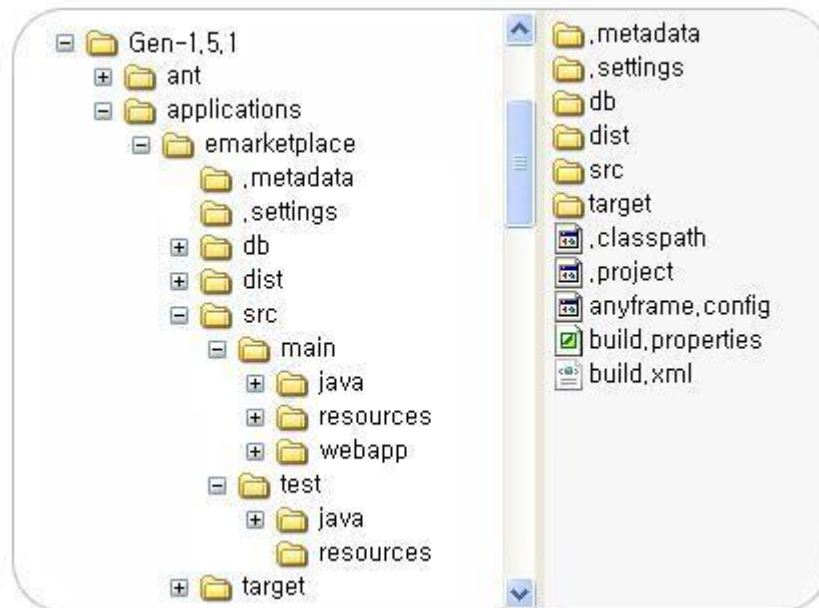
2.2.Anyframe Gen의 구조

Anyframe Gen 설치를 통해 Anyframe, 템플릿, 빌드 도구, 사용되는 모든 라이브러리 설치가 완료된다. 개발 프로젝트의 생성 위치는 꼭 applications 하위로 지정하지 않아도 무방하다.



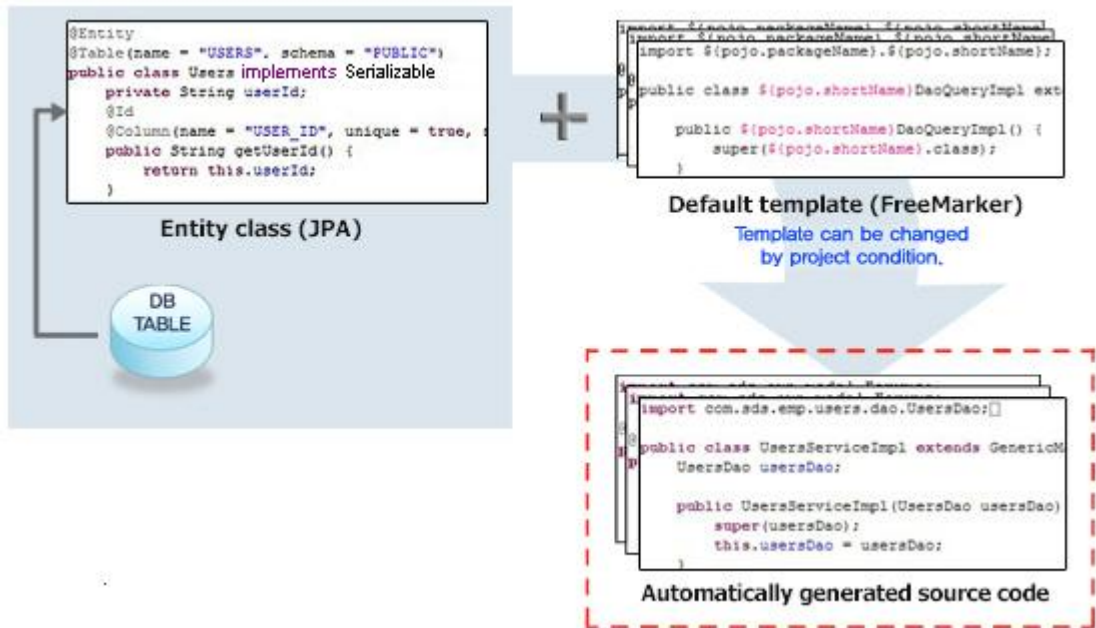
2.3.프로젝트 중심의 개발

프로젝트별 Eclipse 연계 및 샘플 코드 생성이 가능하며 빌드 기능이 제공된다. 어플리케이션 빌드의 경우 클래스 혹은 war 파일로 패키징 처리된다.



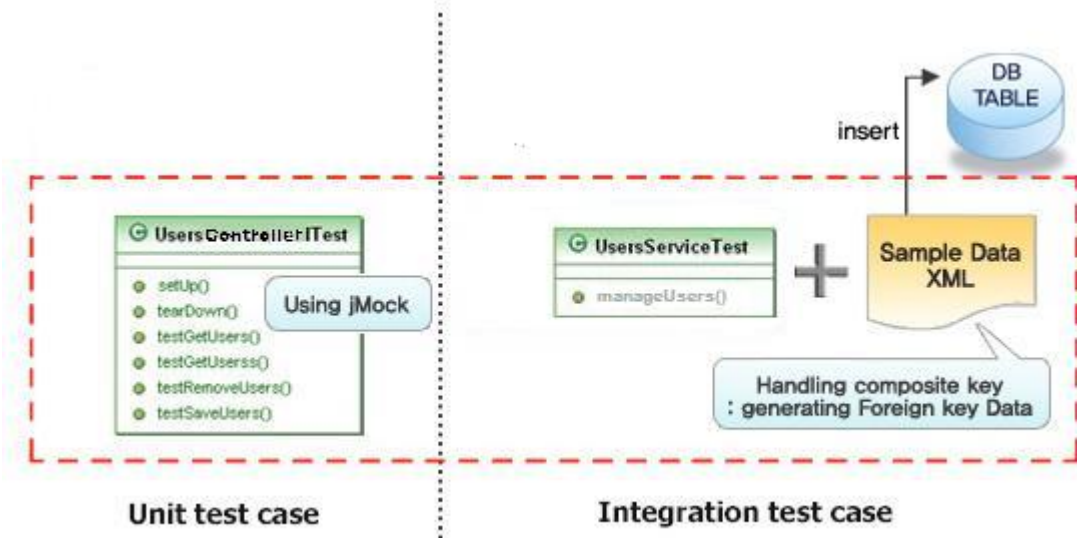
2.4.CRUD 코드 생성

Anyframe Gen을 통해 생성된 도메인 클래스에 대해 CRUD 생성이 가능하다.



2.5.테스트 코드 생성

단위 테스트는 jMock을, 통합 테스트는 JUnit을 사용하여 테스트 하도록 테스트 케이스가 자동생성된다. Foreign Key Data 와 같은 복합 키 데이터 처리 관련한 테스트도 가능하다.



II. Installation

Anyframe Gen을 개발 환경에 맞게 설치하도록 한다. 설치 프로그램을 이용하여 손쉽게 설치할 수 있다.

3.설치

Anyframe Gen은 압축파일로 배포되며, 다음과 같은 순서에 의하여 설치한다. Anyframe Gen 설치 시 Anyframe 도 함께 설치가 된다. 따라서, Anyframe Gen을 설치하면 Anyframe 을 따로 설치할 필요가 없다.

- 1 Anyframe Gen을 설치하기 위한 [시스템 사양]을 확인한다.
- 2 Anyframe Gen의 최신 배포판을 <http://www.anyframejava.org/project/gen#downloads> 에서 확인한다.
- 3 Anyframe Gen의 최신 배포판("Anyframe-gen-x.x.x-bin.zip")을 사용자의 컴퓨터에 내려받는다.
- 4 내려받은 최신 배포판의 압축을 원하는 폴더에 푼다.
- 5 압축을 풀면 다음과 같은 파일들이 보인다.



Name	Size	Type
licenses		Directory
plugins		Directory
Anyframe-gen-1.5.1-setup.jar	37,249KB	Executable Jar File
changelog.txt	44KB	txt
license.txt	12KB	txt
readme.txt	7KB	txt

- licenses : Anyframe Gen을 통해 배포되는 CLI 라이브러리와 Anyframe Repository로 부터 다운받을 수 있는 3rd party 라이브러리들에 대한 라이선스 본문과 정리된 목록
- plugins : Eclipse 플러그인 패키지(Anyframe Common, Anyframe Gen)와 Anyframe Gen Editor에서 Hudson 연계를 위한 파일들(anyframe.ge.eclipse.hudson-X.X.X.jar, web.xml) 포함
- Anyframe-gen-x.x.x-x-setup.jar : Anyframe Gen 설치 파일(JAR 파일을 실행시킨다)
- changelog.txt : 버전 별 변경 사항 로그
- license.txt : Anyframe Gen 라이선스(Apache License)
- readme.txt : Anyframe Gen 소개 및 기본 사항

- 6 Anyframe Gen 설치를 위해 Anyframe-gen-x.x.x-x-setup.jar 파일을 실행시킨다.

jar파일 연결 프로그램이 JAVA Platform 인 경우, 해당 파일을 더블클릭하여 실행시킨다. 만약 알집 등의 압축프로그램으로 연결이 되어 있는 경우는 해당 프로그램의 지원 확장자에서 *.jar를 빼거나, command 창에서 java 명령어를 통해 실행시킬 수 있다. command 창 실행 후, "java -jar Anyframe-gen-x.x.x-x-setup.jar" 명령어를 수행한다.

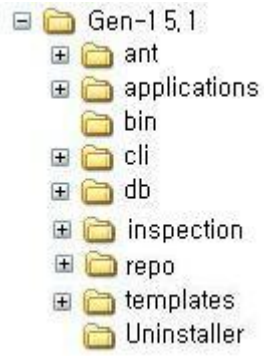
- 7 다음 순서대로 설치를 진행한다.

Apache License에 동의하고 Next 버튼 클릭하면 Anyframe Gen 을 설치할 경로를 지정할 수 있는 창이 조회된다. Anyframe Gen을 설치하고자 하는 폴더를 선택한 후 Next 선택하여 설치를 완료한다. 이때 이 installation path 항목으로 입력된 정보가 **Anyframe Gen01 설치된 root** 경로가 되며 이후 매뉴얼 내용에서 **[Gen Home]** 이라 통칭하도록 한다.

- 8 Anyframe Gen이 정해진 경로에 제대로 설치되어 있는지 확인해본다. 아래 그림과 같은 구조로 설치되어 있으면 정상이다. 설치 과정이 모두 완료되면 Anyframe에서 제공하는 모든 라이브러리들을 다운로드 사용할 수 있는 구조와 빌드 도구인 Ant, 프로젝트 및 코드 생성 템플릿, 공통 빌드 스크립트, 샘플 HSQL DB등이 모두 함께 설치 완료된다.

Anyframe Gen을 통해 배포되는 Ant 빌드도구를 사용하지 않고, Maven을 빌드도구로 사용할 경우, 로컬에 Maven이 설치되어 있어야 한다. Maven은 2.2.1 버전 이상이 설치되어야 한다. (Download Maven site : <http://maven.apache.org/download.html>)

Anyframe Gen 1.5.0부터 [Gen Home]/repo 에 Anyframe에서 제공하는 라이브러리가 포함되지 않고, Anyframe Repository로 부터 다운 받을 수 있도록 settings.xml 파일만 포함한다.



III. Command Line Interface

Command Line Interface를 사용하여 프로젝트 생성, Domain 클래스 및 CRUD 소스 코드 생성, 빌드 및 실행 등의 작업을 할 수 있다. 이때 [빌드]와 [빌드 및 실행]에서 수행하는 빌드가 중복되는 작업이므로 [빌드 및 실행]을 수행시키는 경우 [빌드] 단계는 수행시키지 않는다. CLI(Command Line Interface) 기반의 command를 실행하여 작업하는 경우, 아래 매뉴얼에서 정의한 command arguments 이외의 값 설정 혹은 반드시 필요한 arguments 생략 시 올바르게 동작하지 않을 수 있으므로 유의하도록 한다.

4.환경 설정

Command Line Interface을 사용하기 위한 환경 설정 및 도움말에 대해 알아보자.

4.1.Anyframe Gen 환경 변수 설정

Anyframe Gen 설치 폴더 하위의 bin 폴더에서 command 창을 실행 시킨 다음 env 명령어를 입력하여 env.bat 파일을 실행시킨다.(Windows 환경에서는 env.bat 파일을, Unix 환경에서는 env.sh 파일을 실행시킨다.) GEN_HOME, ANT_HOME, PATH, CLASSPATH, MAINCLASS 등의 환경 변수 설정이 수행된다.



주의 사항

Windows 환경에서는 env.bat 파일을 CLI 상에서 실행시키고, Unix 환경(ex. Mac OS 등)에서는 우선 bin 폴더 하위의 파일들에 대해서 실행 권한을 부여(chmod 755 *)한 후, env.sh 파일을 실행(source env.sh)시킨다. 수행시켜야 할 command는 각 내용의 괄호안을 참고하도록 한다.

4.2.Anyframe Gen 도움말

Command를 통한 작업 시, 사용 가능한 command가 아래 그림과 같으며 이는 gen -help 라는 명령어를 통해 확인 가능하다. gen -help를 통해 확인할 수 있는 명령어는 Anyframe과 Anyframe Gen에서 제공하는 command가 함께 보이는 것이며, Anyframe Gen에서 제공하는 command를 중심으로 확인해본다. Anyframe에서 제공하는 command는 추후 제공되는 Anyframe의 매뉴얼을 참고하도록 한다.

```
C:\WINDOWS\system32\cmd.exe

Welcome to AnyframeGen 1.5.1 - http://anyframejava.org
Licensed under Apache Standard License 2.0
GEN_HOME is set to: C:\Wgen-1.5.1

gen create-project [-options]
. options : -pjtname, -pjtype, -apphome, -package,
            -pjtbuid, -groupid, -artifactid, -version
gen install PLUGIN_NAME [-options]
. options : -package
gen uninstall PLUGIN_NAME [-options]
. options : -excludes
gen list
gen installed
gen install-lib PLUGIN_NAME [-options]
. options : -pjtname, -target, -metadata
gen uninstall-lib PLUGIN_NAME [-options]
. options : -target, -metadata
gen update-catalog [-option]
. option  : -version
gen change-db
gen create-model [-options]
. options : -table, -basepackage
gen create-crud ENTITY [-options]
. options : -package, -scope
gen build [-options]
. options : -deploy, -war, -clean
gen run [-options]
. options : -deploy, -war, -clean
```

4.3.DB 구동

현재 Anyframe Gen은 샘플 DB로 HSQL DB를 제공하고 있다. [Gen Home]\db\scripts\hsqldb\ 하위의 start.cmd 파일을 실행시켜서 DB를 구동한다. 탐색기로 해당 경로 이동 후, start.cmd 파일을 더블 클릭 하면 hsqldb가 구동된다. 만약 Oracle과 같은 다른 DB를 사용한다면 DB를 구동시키고, 프로젝트 생성 후에 DB 정보를 build.properties 파일에 입력해주도록 한다.



주의 사항

Windows 환경에서는 start.cmd 파일을 CLI 상에서 실행시키고, Unix 환경(ex. Mac OS 등)에서는 우선 hsqldb 폴더 하위의 파일들에 대해서 실행 권한을 부여(chmod 755 *)한 후, start.sh 파일을 실행(./start.sh)시킨다.

5.Commands for Project

command를 활용하여 프로젝트 생성, Anyframe 플러그인 설치/제거, Domain Class 및 CRUD 소스 코드 생성, 그리고 DB 정보 변경 등의 작업을 할 수 있다.

5.1.프로젝트 생성

Command Line Interface를 이용해 프로젝트를 생성해본다.

- 1 env를 통해 환경 변수 정보가 설정된 command 창에서 프로젝트 생성을 위해 다음 command를 입력한다. 프로젝트가 생성되는 위치는 command의 옵션에 의해 변경될 수 있다. 여기서는 default인 Gen Home 하위의 applications 폴더에 프로젝트를 생성한다.
- 2 다음의 프로젝트 생성 command를 입력한다.

```
gen create-project [-options]
```

- ex) `gen create-project -pjtname myproject -pjtype web -pjbuild ant -package com.sds.emp`
- 위 예제는 `com.sds.emp` 대표패키지로 하는 `myproject` 프로젝트명의 Ant 빌드 방식 Web 프로젝트를 생성하는 command이다. command로 설정한 정보는 `build.properties` 에 반영되고, Anyframe Basic Archetype 이 설치된다. 아래와 같은 option 정보를 추가하여 프로젝트 정보를 설정할 수 있다.

표 5.1. create-project options

option	description	default value
-pjtname	프로젝트 이름을 지정할 수 있는 옵션으로 이름을 지정하지 않는 경우 default value에 해당하는 프로젝트가 생성된다.	myproject
-pjtttype	web or service 타입 프로젝트를 지정한다. 지정하지 않는 경우 web 타입 프로젝트로 생성된다. *web: dynamic 웹 프로젝트 *service : java 프로젝트	web
-apphome	프로젝트가 생성될 위치를 설정한다. 옵션값을 설정하지 않으면 기본적으로 [Gen Home] 디렉토리 하위의 applications 폴더 아래에 생성된다.	[Gen Home]\applications \
-package	프로젝트의 대표 패키지명을 입력한다. 차후 소스 코드 생성 시 기준이 되는 소스 패키지명이 된다.	프로젝트명으로 -pjtname에 지정한 값
-pjtbuid	프로젝트에서 사용하는 빌드도구를 설정한다. 빌드 방식에 따라, 프로젝트 빌드 스크립트 파일이 아래와 같이 생성된다. "ant"로 설정하는 경우 : build.xml "maven"으로 설정하는 경우 : pom.xml 빌드 스크립트는 프로젝트 생성 후, Anyframe foundation 플러그인을 설치할 때 반영되어 생성된다.	ant
-groupid	Maven 빌드 방식으로 프로젝트를 생성하는 경우, 설정해야 하는 옵션값으로, 프로젝트의 Group Id	com
-artifactid	Maven 빌드 방식으로 프로젝트를 생성하는 경우, 설정해야 하는 옵션값으로, 프로젝트의 Artifact Id	프로젝트명
-version	Maven 빌드 방식으로 프로젝트를 생성하는 경우, 설정해야 하는 옵션값으로, 프로젝트의 Version	1.0-SNAPSHOT

3 프로젝트 생성이 정상적으로 되었는지 확인한다.

생성된 프로젝트는 구조만 있는 상태로, web 타입 프로젝트의 경우 dynamic 웹 프로젝트 구조로, service 타입 프로젝트의 경우 자바 프로젝트 구조로 생성된다.

eclipse 프로젝트 관련 파일들(.project, .classpath)과 빌드 파일(pom.xml), 프로젝트 설정 파일(build.properties), Anyframe 플러그인 정보를 관리하는 폴더(.metadata) 등은 프로젝트 타입과 상관 없이 공통으로 생성된다.

생성된 프로젝트 설정 파일(Properties)을 수정하면, 이후 Domain 클래스 생성 및 CRUD 소스코드 생성 등의 기능에서 수정된 설정 값을 통해 생성되도록 할 수 있다. 아래 표에 설명된 항목별 내용을 보고 DAO Framework, Template type 설정을 비롯하여 다양한 공통 속성 값을 변경시킬 수 있다. DB 관련한 설정 파일 설명에 대해서는 [DB 정보 변경]에 상세하게 설명되어 있다.

표 5.2. build.properties(프로젝트 정보)

Property Name	Description	Required	Default Value
gen.home	Anyframe Gen을 설치한 루트 폴더	Y	[Gen Home]
project.name	프로젝트 명	Y	myproject
project.build	프로젝트 빌드도구	Y	ant
project.version	프로젝트 버전	Y	1.0.0
project.home	프로젝트 루트 폴더 경로	Y	[Gen Home]\applications \\[프로젝트명]
package.name	프로젝트 대표 패키지 명	Y	프로젝트명
dao.framework	DAO Framework 선택(query, hibernate 중 택일)	Y	query
emma.enabled	프로젝트 빌드 시 emma를 이용하여 코드 커버리지 리포트 결과 생성 여부	Y	true
template_type	프로젝트 및 CRUD 소스 코드 생성 시 기반이 되는 Template 형태(ex. default, miplatform 등)	Y	default
project.type	프로젝트 타입	Y	web
web.context.path	프로젝트 빌드도구가 ant일 경우에만 의미있는 값으로, jetty를 이용하여 웹 어플리케이션을 실행할 때 사용되는 WebContext Path명	Y	프로젝트 명

5.2.Anyframe 플러그인 설치 및 제거

프로젝트를 생성한 후, Anyframe 에서 제공하는 다음 command 들을 통해 Anyframe 플러그인을 설치 또는 제거할 수 있다. Anyframe 플러그인은 Foundation 플러그인이 설치된 이후에 정상동작이 가능하므로, 프로젝트를 생성(create-project command 실행)한 후에 Foundation 플러그인을 반드시 설치해야 한다.

service 타입 프로젝트는 플러그인의 라이브러리 파일만 설치/제거 할 수 있는 install-lib, uninstall-lib command 만 사용가능하고, web 타입 프로젝트는 라이브러리 파일 설치 뿐만 아니라 샘플도 함께 설치할 수 있는 command도 모두 사용가능하다. Anyframe 플러그인은 해당 플러그인과 관련된 라이브러리와 샘플로 구성되어 있으며, 샘플은 선택적으로 설치될 수 있다. Ant 빌드 방식의 프로젝트인 경우, install, install-lib command를 실행 할 경우 해당 라이브러리가 src/main/webapp/WEB-INF/lib 에 반영되지만, Maven 빌드 방식의 프로젝트인 경우, pom.xml 에 dependency 정보로 추가된다.

Anyframe Gen CLI를 통해서 생성할 소스코드의 DAO Framework 또는 Template Type을 디폴트로 지정된 값 이외의 값으로 설정하고 싶은 경우, 이와 관련한 Anyframe 플러그인이 설치되어 있어야 생성된 프로젝트가 정상적으로 동작할 수 있다. 예를 들어, DAO Framework의 디폴트 설정은 Query Service로 되어 있지만 Hibernate/JPA 를 사용하고 싶은 경우, 프로젝트 설정 파일(build.properties)의 dao.framework 의 값을 hibernate으로 변경하는 것 이외에, Anyframe 의 hibernate 플러그인을 설치하여야 한다. 프로젝트 설정 파일과 관련한 설명은 [프로젝트 생성]을 참조하도록 한다.

- Anyframe 플러그인(라이브러리+샘플코드) 설치
 - 1 프로젝트가 생성된 폴더 하위로 이동한다. (예를 들어, myproject 라는 프로젝트를 생성시켰다면 myproject 폴더 하위에서 command를 실행시키도록 한다.)
 - 2 Anyframe 플러그인 설치(라이브러리+샘플코드) command를 입력한다.

```
gen install PLUGIN_NAME [-options]
```

- ex) gen install hibernate -package com.sds.emp

- 위 예제는 hibernate 플러그인에 대한 샘플과 라이브러리를 설치하는 command로 설치되는 샘플의 패키지는 com.sds.emp로 설정된다.

표 5.3. install options

option	description	default value
-package	플러그인 설치 시 함께 설치되는 샘플 소스 코드의 패키지를 지정하는 옵션으로 지정하지 않았을 경우 src/main/java 밑에 상위패키지 없이 설치된다.	N/A

- 3 Anyframe 플러그인(라이브러리+샘플코드)이 정상적으로 설치되었는지 확인한다.

Anyframe 플러그인(라이브러리+샘플코드)이 정상적으로 설치되었다면, 지정한 패키지 경로에 설치한 플러그인 샘플 소스 코드가 생성되어 있고, src/main/webapp/WEB-INF/lib에 플러그인 관련 라이브러리가 설치되어 있다. Maven 프로젝트의 경우 pom.xml 에 플러그인 관련 라이브러리에 대한 dependency 가 설정된다.

표 5.4. result

	web 타입 프로젝트	service 타입 프로젝트
ant 빌드	지정한 패키지 경로에 설치한 플러그인 샘플 소스 코드 생성, src/main/webapp/WEB-INF/lib에 플러그인 관련 라이브러리 설치	-(지원되지 않음)
maven 빌드	pom.xml 에 플러그인 관련 dependency 추가	-(지원되지 않음)

- Anyframe 플러그인(라이브러리+샘플코드) 제거
 - 1 프로젝트가 생성된 폴더 하위로 이동한다. (예를 들어, myproject라는 프로젝트를 생성시켰다면 myproject 폴더 하위에서 command를 실행시키도록 한다.)
 - 2 Anyframe 플러그인 제거(라이브러리+샘플코드) command를 입력한다.

```
gen uninstall PLUGIN_NAME [-options]
```

- ex) gen uninstall hibernate
- 위 예제는 hibernate 플러그인에 대한 샘플과 라이브러리를 제거하는 command이다.

표 5.5. uninstall options

option	description	default value
-excludes	플러그인 제거시 제외하고 싶은 파일명을 입력한다. 예를 들어, context-hibernate-services.xml 을 hibernate 플러그인 삭제시 삭제되지 않게 하기 위해서는 -excludes context-hibernate-services.xml 로 옵션을 설정해준다.	N/A

- 3 Anyframe 플러그인(라이브러리+샘플코드)이 정상적으로 제거되었는지 확인한다.

Anyframe 플러그인(라이브러리+샘플코드)이 정상적으로 제거되었다면, 플러그인 샘플 소스 코드 및 관련 라이브러리가 제거된다.

표 5.6. result

	web 타입 프로젝트	service 타입 프로젝트
ant 빌드	지정한 패키지 경로에 설치한 플러그인 샘플 소스 코드 삭제, src/main/webapp/WEB-INF/lib에 플러그인 관련 라이브러리 설치	-(지원되지 않음)
maven 빌드	지정한 패키지 경로에 설치한 플러그인 샘플 소스 코드 삭제, pom.xml 에 플러그인 관련 dependency 삭제	-(지원되지 않음)

- Anyframe 플러그인(라이브러리) 설치

- 1 프로젝트가 생성된 폴더 하위로 이동한다. (예를 들어, myproject라는 프로젝트를 생성시켰다면 myproject 폴더 하위에서 command를 실행시키도록 한다.)
- 2 Anyframe 플러그인(라이브러리) 설치 command를 입력한다.

```
gen install-lib PLUGIN_NAME [-options]
```

- ex) gen install-lib hibernate
- 위 예제는 hibernate 플러그인에 대한 라이브러리를 설치하는 command이다.

표 5.7. install-lib options

option	description	default value
-pjtname	라이브러리 설치 할 프로젝트명을 지정하는 옵션이다.	command를 실행시키는 위치의 프로젝트명
-target	설치할 라이브러리가 위치할 파일 경로를 지정하는 옵션이다.	[프로젝트 폴더]/src/main/webapp/WEB-INF/lib
-metadata	플러그인 정보가 있는 metadata 파일의 위치를 지정하는 옵션이다.	프로젝트 폴더 하위

- 3 Anyframe 플러그인(라이브러리)이 정상적으로 설치되었는지 확인한다.

Anyframe 플러그인(라이브러리)이 정상적으로 설치되었다면, 아래와 같이 라이브러리가 반영된다.

표 5.8. result

	web 타입 프로젝트	service 타입 프로젝트
ant 빌드	src/main/webapp/WEB-INF/lib에 플러그인 관련 라이브러리 설치	프로젝트가 생성된 위치 하위의 lib 폴더에 플러그인 관련 라이브러리 설치
maven 빌드	pom.xml 에 플러그인 관련 dependency 추가	pom.xml 에 플러그인 관련 dependency 추가

- Anyframe 플러그인(라이브러리) 제거

- 1 프로젝트가 생성된 폴더 하위로 이동한다. (예를 들어, myproject라는 프로젝트를 생성시켰다면 myproject 폴더 하위에서 command를 실행시키도록 한다.)
- 2 Anyframe 플러그인(라이브러리) 제거 command를 입력한다.

```
gen uninstall-lib PLUGIN_NAME [-options]
```

- ex) gen uninstall-lib hibernate
- 위 예제는 hibernate 플러그인에 대한 라이브러리를 제거하는 command이다.

표 5.9. uninstall-lib options

option	description	default value
-target	삭제할 라이브러리가 위치한 파일 경로를 지정하는 옵션이다.	[프로젝트 생성 폴더]/src/main/webapp/WEB-INF/lib
-metadata	플러그인 정보가 있는 metadata 파일의 위치를 지정하는 옵션이다.	프로젝트 생성 폴더 하위

- 3 Anyframe 플러그인(라이브러리)이 정상적으로 제거되었는지 확인한다.

Anyframe 플러그인(라이브러리)이 정상적으로 제거되었다면, 플러그인 관련 라이브러리가 제거된다.

표 5.10. result

	web 타입 프로젝트	service 타입 프로젝트
ant 빌드	src/main/webapp/WEB-INF/lib에 플러그인 관련 라이브러리 삭제	프로젝트가 생성된 위치 하위의 lib 폴더에 플러그인 관련 라이브러리 삭제
maven 빌드	pom.xml 에 플러그인 관련 dependency 삭제	pom.xml 에 플러그인 관련 dependency 삭제

5.3.Anyframe 플러그인 목록 조회 및 업데이트

Anyframe 플러그인 목록을 조회하고, 업데이트한다.

Anyframe 플러그인 목록은 settings.xml 파일이 존재하는 폴더에 plugin-catalog-X.X.X.xml 파일(X.X.X은 버전정보) 형태로 관리되며, Anyframe 플러그인 목록 조회 및 업데이트 command를 실행하면, plugin-catalog-X.X.X.xml 파일을 기준으로 조회, 업데이트 된다.

- Anyframe 플러그인 목록 조회
 - 1 프로젝트가 생성된 폴더 하위로 이동한다. (예를 들어, myproject 라는 프로젝트를 생성시켰다면 myproject 폴더 하위에서 command를 실행시키도록 한다.)
 - 2 Anyframe 플러그인 목록 조회 command를 입력한다.

```
gen list
```

- ex) gen list
- 위 예제는 Anyframe 플러그인 목록을 조회한다.

3 Anyframe 플러그인 목록을 조회한다.

- Anyframe 플러그인 설치된 목록 조회

1 프로젝트가 생성된 폴더 하위로 이동한다. (예를 들어, myproject 라는 프로젝트를 생성시켰다면 myproject 폴더 하위에서 command를 실행시키도록 한다.)

2 Anyframe 플러그인 설치된 목록 업데이트 command를 입력한다.

```
gen installed
```

- ex) gen installed
- 위 예제는 설치된 Anyframe 플러그인 목록을 조회한다.

3 설치된 Anyframe 플러그인 목록을 조회한다.

- Anyframe 플러그인 목록 업데이트

1 프로젝트가 생성된 폴더 하위로 이동한다. (예를 들어, myproject 라는 프로젝트를 생성시켰다면 myproject 폴더 하위에서 command를 실행시키도록 한다.)

2 Anyframe 플러그인 목록 업데이트 command를 입력한다.

```
gen update-catalog [-option]
```

- ex) gen update-catalog
- 위 예제는 Anyframe Repository로 부터 Anyframe 플러그인 목록 파일(plugin-catalog-X.X.X.xml)을 다운로드 받는다.

표 5.11. update-catalog options

option	description	default value
-version	플러그인 목록 파일 버전정보	[Gen Home]\cli\lib\anyframe.command.common-X.X.X.jar에 있는 plugins.xml의 foundation 플러그인 version

3 Anyframe 플러그인 목록이 업데이트된다.

5.4.Domain Class 생성

Command Line Interface를 이용해 Domain Class를 생성해본다. 이때, 반드시 DB를 구동하여 다음 command를 수행한다. DB가 구동되지 않은 상태에서 Domain Class를 생성하는 경우 socket creation

error가 발생한다. 기본적으로 DB 정보는 HSQL로 설정되나 이를 변경하고 싶은 경우 다음 [DB정보 변경] 을 참조한다.

- 1 프로젝트가 생성된 폴더 하위로 이동한다. (예를 들어, myproject 라는 프로젝트를 생성시켰다면 myproject 폴더 하위에서 command를 실행시키도록 한다.)
- 2 Domain Class 생성 command를 입력한다.

```
gen create-model [-options]
```

- ex) gen create-model -table "*" -basepackage com.sds.emp.domain
- 위 예제는 모든 테이블에 대해 Domain Class를 프로젝트 생성 시 지정한 대표 패키지(ex. com.sds.emp) 하위의 domain 패키지 내에 생성하는 command로 아래와 같은 option 정보를 이용하여 Domain Class 생성 대상 및 패키지 정보를 변경할 수 있다.

표 5.12. create-model options

option	description	default value
-table	<p>Domain Class를 생성하려는 대상 DB 테이블을 선정한다. 테이블에 대한 option 을 따로 설정하지 않는 경우 default로 모든 DB 테이블로부터 Domain Class를 생성하고 있다. 명시적으로 모든 DB Table에 대해서 Domain Class를 생성하기 위해서는 "*" 로 옵션 값을 설정한다. 특정 테이블만 대상으로 Domain Class를 생성하고 싶은 경우 대상 DB 테이블명을 지정하면 된다. 여러 개의 테이블을 대상으로 할 때에는 "," 기호를 이용하여 작성하도록 한다.</p> <p>* 이때 테이블명에 대해 대소문자를 구분하므로 유의하여 테이블명을 입력하도록 한다. 특히, DB에 맞게 대소문자를 표현해야 한다. HSQLDB, Oracle의 경우 대문자로 테이블명을 표현해야 인식하지만, MySQL의 경우 테이블명을 소문자로 표현해야한다.</p> <p>ex) gen create-model -table BOARD,BOARD_MASTER -basepackage com.sds.emp.domain</p>	"*" (모든 테이블 대상)
-basepackage	DB 테이블로부터 Domain Class를 생성 시 어느 소스 코드 패키지 하위에 생성시킬 지 결정하는 옵션으로, default로 프로젝트 생성 시 설정한 대표패키지 + ".domain" 패키지 경로가 Domain Class 패키지 경로가 된다.	프로젝트 생성 시 지정한 대표 패키지.domain

- 3 Domain Class 가 정상적으로 생성되었는지 확인한다.

Domain Class 생성 시 지정한 패키지에, camelcase 적용된 테이블 이름의 도메인 클래스가 있으면 정상이다. 아래는 생성된 Board.java 코드의 일부분이다.

```
@Entity
@Table(name = "BOARD", schema = "PUBLIC")
public class Board implements Serializable {
    private BoardId id;
    private BoardMaster boardMaster;
    private String boardName;
    중략...
    public BoardId getId() {
        return this.id;
    }
    public void setId(BoardId id) {
        this.id = id;
    }
}
```

```

@ManyToOne(fetch = FetchType.LAZY)
@JoinColumn(name = "BOARD_MASTER_ID", nullable = false,
            insertable = false, updatable = false)
public BoardMaster getBoardMaster() {
    return this.boardMaster;
}
public void setBoardMaster(BoardMaster boardMaster) {
    this.boardMaster = boardMaster;
}
@Column(name = "BOARD_NAME", nullable = false, length = 150)
public String getBoardName() {
    return this.boardName;
}
public void setBoardName(String boardName) {
    this.boardName = boardName;
}
}
}

```

5.5.CRUD 소스 코드 생성

위에서 생성한 Domain Class를 기반으로 CRUD 소스 코드를 생성할 수 있다. 이때 Business Layer와 Presentation Layer 별로 구분하여 소스 코드를 생성할 수도 있고 함께 생성해낼 수도 있다. 각각의 경우로 나누어서 아래에 설명하고 있다.

Maven 빌드방식의 프로젝트의 경우, CRUD 소스 코드를 생성하기 전에 Domain 클래스에 대해 컴파일 이 선행되어야 한다. Maven 컴파일은 command창에서 다음과 같은 명령어를 통해 수행가능하다. 이 command는 프로젝트가 생성된 폴더 하위에서 실행하도록 한다.

```
mvn compile
```

아래 예제는 default 템플릿 타입으로 설정되어 있는 경우에 해당하며, 이 때 생성된 CRUD 소스코드는 annotation 기반으로 되어 있다.

5.5.1.Business Layer 코드 생성

- 1 프로젝트가 생성된 폴더 하위로 이동한다. (예를 들어, myproject 라는 프로젝트를 생성시켰다면 myproject 폴더 하위에서 command를 실행시키도록 한다.)
- 2 Business Layer 코드 생성 command를 입력한다.

```
gen create-crud ENTITY [-options]
```

- ex) gen create-crud Board -package board **-scope service**
- 위 예제는 Board Domain class를 기반으로 CRUD 소스 코드 생성 시 대표 패키지(ex. com.sds.emp) + ".board" 패키지 하위에 business layer 코드를 생성시키는 command로 아래와 같은 option 정보를 이용하여 CRUD 소스 코드 생성 시 필요한 타겟 프로젝트와 패키지 경로 위치 그리고 생성 범위 등을 변경할 수 있다.

표 5.13. create-crud options

option	description	default value
-package	CRUD 소스 코드가 생성될 패키지명을 정하는 옵션으로, 프로젝트 생성 시 지정한 대표 패키지 뒤에 붙게 된다.	Domain Class명(소문자)형태
-scope	생성할 코드가 business layer일 경우 service로 설정하고, presentation layer일 경우 web으로 설정한다. 두 layer에 대한 소스 코드 모두를 함께 생성할 경우 all로 설정한다.	all

3 Business Layer 코드가 정상적으로 생성되었는지 확인한다.

서비스인터페이스, 서비스구현클래스, DAO구현클래스와 매핑쿼리문등이 생성된다.

아래는 생성된 자바코드와 속성파일, 매핑쿼리문의 일부분이다.

- **서비스인터페이스** (src/main/java/[대표패키지]/[-package 값]/service/[Entity 클래스명]Service.java)

```
public interface BoardService{
    void create(Board board) throws Exception;
    void remove(BoardId id) throws Exception;
    중략...
}
```

- **서비스구현클래스** (src/main/java/[대표패키지]/[-package 값]/service/impl/[Entity 클래스명]ServiceImpl.java)

```
@Service("boardService")
@Transactional(rollbackFor = {Exception.class}, propagation = Propagation.REQUIRED)
public class BoardServiceImpl implements BoardService {

    @Inject
    @Named("boardDao")
    private BoardDao boardDao;

    public void create(Board board) throws Exception {
        this.boardDao.create(board);
    }

    public void remove(BoardId id) throws Exception {
        this.boardDao.remove(id);
    }
    중략...
}
```

- **DAO클래스** (src/main/java/[대표패키지]/[-package 값]/service/impl/[Entity 클래스명]Dao.java)

```
@Repository("boardDao")
public class BoardDaoQueryImpl extends AbstractDAO {

    @Value("#{contextProperties['pageSize'] ?: 10}")
    int pageSize;

    @Value("#{contextProperties['pageUnit'] ?: 10}")
    int pageUnit;

    @Inject
    public void setQueryService(IQueryService queryService){
```

```

        super.setQueryService(queryService);
    }

    /** {@inheritDoc} */
    public void create(Board board) throws Exception {
        super.create("Board",board);
    }

    /** {@inheritDoc} */
    public void remove(BoardId id) throws Exception {
        Board board = new Board();
        board.setId(id);
        super.remove("Board",board);
    }
    중략...
}

```

- 매핑쿼리문 (src/main/resources/sql/mapping-query-[Entity 클래스명].xml)

```

<queryservice>
  <queries>
    <query id="createBoard">
      <statement>
        INSERT INTO BOARD (BOARD_ID, BOARD_DESC, BOARD_MASTER_ID, BOARD_NAME,
          BOARD_ORDER, BOARD_TOPICS, REG_DATE)
        VALUES (:vo.id.boardId, :vo.boardDesc, :vo.id.boardMasterId, :vo.boardName,
          :vo.boardOrder, :vo.boardTopics, :vo.regDate)
      </statement>
    </query>
    중략...
  </queries>
</queryservice>

```

- MessageSource (src/main/resources/message/message-generation.properties)

```

# -- Board-START
board.id.boardId=Board Id
board.id.boardMasterId=Board Master Id
board.boardDesc=Board Desc
board.boardName=Board Name
board.boardOrder=Board Order
board.boardTopics=Board Topics
board.regDate=Reg Date

# -- success messages --
success.board.create=Board has been added successfully.
success.board.update=Board has been updated successfully.
success.board.delete=Board has been deleted successfully.

# -- error messages --
error.boardserviceimpl.create=Board data not created
error.boardserviceimpl.create.solution=Enter correct data for mandatory field
or enter data according to formats means date format as yyyy-mm-dd
error.boardserviceimpl.create.reason=Entered incorrect data for Board
중략...

```

- 통합테스트케이스 (src/test/java/[대표패키지]/[-package 값]/service/[Entity 클래스명]ServiceTest.java)

```

@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(locations = { "file:./src/main/resources/spring/context-*.xml" })

```

```
public class BoardServiceTest{

    @Inject
    @Named("boardDao")
    private BoardDao boardDao;

    @Test
    @Rollback(value=true)
    public void manageBoard() throws Exception {
        //1. create a new board
        Board board = getBoard();

        if(boardService.get(board.getId()) != null)
            boardService.remove(board.getId());

        boardService.create(board);
        중략...
    }
    중략...
}
```



참고

build.properties 에서 dao.framework 값을 변경하지 않았을 경우 기본 DAO Framework 은 Query Service 로 설정되어 있다. 위의 경우는 Query Service로 설정되어 있는 경우에 해당한다.

build.properties 에서 dao.framework 값을 hibernate 으로 설정하여 DAO Framework 으로 Hibernate 을 사용하는 경우, CRUD 소스코드 생성시 DAO구현 클래스는 BoardDao.java 로, 매핑쿼리문은 hibernate/dynamic-hibernate-board.xml 형태로 생성된다.

5.5.2.Presentation Layer 코드 생성

- 1 프로젝트가 생성된 폴더 하위로 이동한다. (예를 들어, myproject 라는 프로젝트를 생성시켰다면 myproject 폴더 하위에서 command를 실행시키도록 한다.)
- 2 Presentation Layer 코드 생성 command를 입력한다.

```
gen create-crud ENTITY [-options]
```

- ex) gen create-crud Board -package board -scope web
- 위 예제는 Board Domain class를 기반으로 CRUD 소스 코드 생성 시 대표 패키지(ex. com.sds.emp) + ".board" 패키지 하위에 presentation layer 코드를 생성시키는 command이다.

표 5.14. create-crud options

option	description	default value
-package	CRUD 소스 코드가 생성될 패키지명을 정하는 옵션으로, 프로젝트 생성 시 지정한 대표 패키지 뒤에 붙게 된다.	Domain Class명(소문자)형태
-scope	생성할 코드가 business layer일 경우 service로 설정하고, presentation layer일 경우 web으로 설정한다. 두 layer에 대한 소스 코드 모두를 함께 생성할 경우 all로 설정한다.	all

- 3 Presentation Layer 코드가 정상적으로 생성되었는지 확인한다.

서비스와 통신을 하기 위한 컨트롤러 파일, 비파일(JSP 또는 MiPlatform 용 XML파일) 등이 생성된다.

- 컨트롤러 (src/main/java/[대표패키지]/[-package 값]/web/[Entity 클래스명]Controller.java)

```

@Controller
@RequestMapping("/board.do")
public class BoardController {

    /**
     * Resource Injection on BoardService
     */
    @Inject
    @Named("boardService")
    private BoardService boardService;

    중략...

    /**
     * Display form for adding Board.
     * @param model model containing control data
     * @return the prepared form view
     * @throws Exception in case of an invalid new form object
     */
    @RequestMapping(params = "method=createView")
    public String createView(Model model) throws Exception
    {
        model.addAttribute(new Board());

        return "genViewBoard";
    }
}

```

- UI JSP 파일 (src/main/webapp/WEB-INF/jsp/generation/[Entity 클래스명]/*.jsp)

```

<%@ page language="java" errorPage="/sample/common/error.jsp"
      pageEncoding="UTF-8" contentType="text/html; charset=utf-8" %>

<%@ include file="/sample/common/taglibs.jsp"%>
<html>
<head>
    <%@ include file="/sample/common/meta.jsp" %>
    <title><fmt:message key="boardList.title"/></title>
    <meta name="heading" content="<spring:message code='boardList.heading'/'>"/>
    <link rel="stylesheet" href="<c:url value='/sample/css/admin.css'/'>"
          type="text/css">

    <script type="text/javascript"
            src="<c:url value='/sample/javascript/CommonScript.js'/'>"></script>
    <script language="JavaScript">
        function fncCreateBoardView() {
            document.location.href="<c:url value='/board.do?method=createView'/'>";
        }
    }
    중략...

```

- 단위테스트케이스 (src/test/java/[대표패키지]/[-package 값]/web/[Entity 클래스명]ControllerTest.java)

```

@RunWith(JMock.class)
public class BoardControllerTest {

    private BoardController controller;
    private String SUCCESS_CREATEVIEW = "genViewBoard";
}

```

```

private String SUCCESS_CREATE = "redirect:/board.do?method=list";
private String SUCCESS_GET = "genViewBoard";
private String SUCCESS_UPDATE = "redirect:/board.do?method=list";
private String SUCCESS_LIST = "genListBoard";
private String SUCCESS_REMOVE = "redirect:/board.do?method=list";
private Mockery context = new JUnit4Mockery();
private BoardService mockService = null;

@Before
public void setUp() throws Exception {
    System.setProperty("log4j.configuration", "log4j-test.xml");

    this.mockService = context.mock(BoardService.class);
    this.controller = new BoardController();
    this.controller.setBoardService(this.mockService);
}

public void setBoardController(BoardController controller) {
    this.controller = controller;
}

@Test
public void testCreateView() throws Exception{
    String viewName = this.controller.createView(new ExtendedModelMap());

    assertEquals("returned correct view name", SUCCESS_CREATEVIEW, viewName);
}
중략...

```

5.5.3. Business Layer와 Presentation Layer 코드 함께 생성

- 1 프로젝트가 생성된 폴더 하위로 이동한다. (예를 들어, myproject 라는 프로젝트를 생성시켰다면 myproject 폴더 하위에서 command를 실행시키도록 한다.)
- 2 Business Layer 와 Presentation Layer 코드 생성 command를 입력한다.

```
gen create-crud ENTITY [-options]
```

- ex) gen create-crud Board -package board -scope all
- 위 예제는 Board Domain class를 기반으로 CRUD 소스 코드 생성 시 대표 패키지(ex. com.sds.emp) + ".board" 패키지 하위에 business layer 코드와 presentation layer 코드를 함께 생성시키는 command이다.

표 5.15. create-crud options

option	description	default value
-package	CRUD 소스 코드가 생성될 패키지명을 정하는 옵션으로, 프로젝트 생성 시 지정한 대표 패키지 뒤에 붙게 된다.	Domain Class명(소문자)형태
-scope	생성할 코드가 business layer일 경우 service로 설정하고, presentation layer일 경우 web으로 설정한다. 두 layer에 대한 소스 코드 모두를 함께 생성할 경우 all로 설정한다.	all

- 3 Business Layer, Presentation Layer 코드가 정상적으로 생성되었는지 확인한다. dao,service,web 폴더가 존재하여야 하며, 각각 business layer, presentation layer 생성 시 결과 화면을 참조하여 확인해보도록 한다.



주의 사항

service 타입일 때는 "gen install-lib" command를 통해 라이브러리만 설치되기 때문에, hibernate, miplatform 관련 spring 설정파일(context*.xml)이 설치되지 않는다. 따라서, dao framework을 hibernate으로 설정하거나, template type을 miplatform으로 선택하여 소스코드를 생성한 경우, 자동생성된 테스트 케이스는 수행시 오류가 발생하게 된다. Fail to ApplicationContext...에러가 발생하면 hibernate 또는 miplatform spring 설정파일을 설치하도록한다.

5.6.DB 정보 변경

Command Line Interface를 이용하여 build.properties 파일에 변경한 DB 설정 정보를 프로젝트 내 관련 파일들에 모두 반영되도록 한다. default로 HSQL DB를 사용하는 sampledb를 제공하고 있다. 해당 프로젝트에서 HSQL DB가 아닌 다른 DB를 사용하는 경우 build.properties 파일을 수정해준 뒤, DB 정보 변경 command를 실행시켜준다.

- 1 build.properties파일(생성한 프로젝트 폴더 하위에 위치)을 열어 DB 설정 정보를 수정한다.

표 5.16. build.properties I (DB 정보)

property name	description	required	default value
db_type	DB 정보(HSQL, Oracle, MySQL, Sybase 등)	Y	HSQL
db_name	Database 명	Y	sampledb
schema	schema 명	N	PUBLIC
username	DB 사용자 명	Y	sa
password	DB 사용자 패스워드	N	
server	DB에 접근하기 위한 server 정보 (ex.70.7.105.123)	Y	localhost
port	DB에 접근하기 위한 port 번호	Y	-1
driver_class	DB의 드라이버 클래스를 설정한다. Default Value로는 HSQL의 드라이버 클래스 값이며, DB에 맞는 값을 찾아 설정하면 된다. ex. Oracle: oracle.jdbc.driver.OracleDriver ex. MySQL: com.mysql.jdbc.Driver ex. Sybase: com.sybase.jdbc3.jdbc.SybDataSource	Y	org.hsqldb.jdbcDriver
driver_jar_path	DB에 접근하기 위한 driver jar 파일 경로를 설정한다. Default Value로는 HSQL의 driver jar파일 경로 값이며, DB에 맞는 driver jar 파일을 찾아 경로를 설정하면 된다. 이때, 패스 표현 방법에 주의를 기울여야 한다. 상대 경로로 입력시 경로 처음에 '/'로 시작하게 될 경우 절대경로로 인식하게된다. ex. Oracle: ojdbc-XX.jar ex. MySQL: mysql-connector-java-X.X.XX-bin.jar ex. Sybase: jconnX.jar	Y	[Gen Home]/db/ lib/ hsqldb- 1.8.0.10.jar
url	DB에 접근하기 위한 Access URL	Y	jdbc: hsqldb:hsqldb: // localhost /sampledb

표 5.17. build.properties II (DB 관련 정보)

property name	description	required	default value
dialect	<p>DB별 Hibernate Dialect 클래스를 설정한다. Default Value로는 HSQL의 dialect값이며, DB에 맞는 값을 찾아 설정하면 된다.</p> <p>ex. Oracle: org.hibernate.dialect.OracleDialect(10g의 경우 org.hibernate.dialect.Oracle10gDialect)</p> <p>ex. MySQL: org.hibernate.dialect.MySQLDialect</p> <p>ex. Sybase: org.hibernate.dialect.SybaseDialect</p>	Y	org.hibernate.dialect.HSQLDialect
paging_generator	<p>[Query Service에서 사용] 목록 조회 시 페이징 처리를 지원하기 위한 SQL 생성 클래스명을 설정한다. Default Value로는 HSQL의 paging generator 값이며, DB에 맞는 값을 설정하면 된다. paging generator 값이 설정되어야 Query Service 사용이 가능하다. Sybase의 경우 DB에 특화된 generator는 없고, default paging sql generator를 사용한다.</p> <p>ex. Oracle: anyframe.core.query.impl.jdbc.generator.OraclePagingSQLGenerator</p> <p>ex. MySQL: anyframe.core.query.impl.jdbc.generator.MySQLPagingSQLGenerator</p> <p>ex. Sybase: anyframe.core.query.impl.jdbc.generator.DefaultPagingSQLGenerator</p>	Y	anyframe.core.query.impl.jdbc.generator.HSQLPagingSQLGenerator
lobhandler	<p>[Query Service에서 사용] BLOB,CLOB과 같은 Large Data Type을 핸들링하는 클래스명을 설정한다. Default Value로는 HSQL의 lobhandler 값이며, HSQL은 lob type을 지원하지 않으므로 공란으로 비워둔다. 각 DB에 맞는 lobhandler 값을 찾아 설정한다.</p> <p>ex. Oracle: org.springframework.jdbc.support.lob.OracleLobHandler</p> <p>ex. MySQL: org.springframework.jdbc.support.lob.DefaultLobHandler</p> <p>ex. Sybase: org.springframework.jdbc.support.lob.DefaultLobHandler</p>	N	N/A

2 DB 변경 command를 입력한다. build.properties 파일의 DB 정보를 참조로 하여 DB 설정 내용을 반영한다.

```
gen change-db
```

3 DB 정보가 정상적으로 변경되었는지 확인한다.

프로젝트 코드 내 DataSource Service 설정 정보 및 Hibernate, Query Service 관련 정보들이 변경되었는지 확인한다. (ex. src/main/resources/spring/context-datasource.xml)



참고

현재 Anyframe Gen에서는 5개의 DB를 지원하고 있다.(HSQL, Oracle, MySQL, Sybase, DB2) 프로젝트에서 작업하려는 DB가 이 5가지 중 하나라면 [Gen Home]/db/jdbc.config 파일을 열어서 사용하려는 DB 정보가 등록되었는지 확인한다. ※ Gen 이클립스 플러그인 UI에서는 4개의 DB 목록(HSQL, Oracle, MySQL, Sybase)을 제공하고 있다.

```
<list>
<jdbcType>
<type>Oracle</type>
<driver>oracle.jdbc.driver.OracleDriver</driver>
<dialect>
<string>org.hibernate.dialect.Oracle10gDialect</string>
<string>org.hibernate.dialect.Oracle9iDialect</string>
<string>org.hibernate.dialect.Oracle8iDialect</string>
</dialect>
<port>1521</port>
<pagingGenerator>anyframe.core.query.impl.jdbc.generator.
OraclePagingSQLGenerator</pagingGenerator>
<lobHandler>org.springframework.jdbc.support.lob.OracleLobHandler
</lobHandler>
</jdbcType>
중략...
```

만약 이외의 다른 DB를 사용하려면 Anyframe 이슈 관리 시스템 [<http://dev.anyframejava.org/jira/>] 에 등록해주시기 바랍니다.

6. 어플리케이션 및 프로젝트 빌드

Command Line Interface를 이용해 생성된 프로젝트에서 어플리케이션 및 프로젝트에 대해 빌드를 할 수 있다. 빌드 단계에서는 소스 코드 컴파일, 테스트 케이스 수행, 패키징, 코드 품질 검사, 배포 파일 생성 등 빌드 프로세스를 수행하는데 어플리케이션 빌드와 프로젝트 빌드 두 가지 형태로 크게 구분할 수 있다.

프로젝트 단위로 빌드를 수행하여 프로젝트 배포 파일을 생성해낸다.

6.1. 어플리케이션 빌드

Command Line Interface를 이용해 어플리케이션을 빌드해본다. 이 command는 프로젝트가 Ant 를 이용하여 빌드하는 web 타입인 경우에만 이용가능하다.

- 1 프로젝트가 생성된 폴더 하위로 이동한다. (예를 들어, myproject 라는 프로젝트를 생성시켰다면 myproject 폴더 하위에서 command를 실행시키도록 한다.)
- 2 다음의 어플리케이션 빌드 command를 입력한다.

```
gen build [-options]
```

- 프로젝트가 생성된 폴더 하위의 dist폴더에 프로젝트 명으로 된 배포폴더가 생성되고, 옵션에 따라 배포형태 및 war 파일 생성유무 등이 결정된다.
- ex)gen build -deploy jar -war
- 위 예제는 jar 파일 형태로 프로젝트를 패키징하여 배포 파일을 생성하고, war 파일을 생성하는 command이다.

표 6.1. build options

option	description	default value
-deploy	프로젝트 배포 형태를 정의할 수 있다. "class" 혹은 "jar"를 입력한다. - WEB-INF/classes 폴더 하위에 클래스 형태로 배포할 때 class로 설정 - WEB-INF/lib 폴더 하위에 jar 파일 형태로 배포할 때 jar로 설정	class
-war	-war 를 쓰면 프로젝트가 생성된 폴더 하위의 dist 폴더에 프로젝트 명.war 파일을 생성 한다.	N/A
-clean	-clean 을 쓰면 프로젝트가 생성된 폴더 하위의 dist 폴더 전체를 삭제 한 후 빌드를 수행한다.	N/A

6.1.1. 예시1) 어플리케이션 빌드 후 클래스 배포

어플리케이션 전체 빌드 후, WEB-INF/classes 폴더 하위에 프로젝트 소스 코드 컴파일 결과를 class 파일 형태로 배포하는 경우에 사용한다.

```
gen build -deploy class
```

빌드된 결과를 WEB-INF/classes 에서 확인하며, WEB-INF/classes 아래에 컴파일된 자바코드, 설정파일, 매핑쿼리문 등이 생성된다.

6.1.2.예시2) 어플리케이션 빌드 후 JAR 파일 배포

어플리케이션 전체 빌드 후, WEB-INF/lib 폴더 하위에 프로젝트 소스 코드 컴파일 결과를 jar 파일 형태로 배포하는 경우에 사용한다.

```
gen build -deploy jar
```

빌드된 결과를 WEB-INF/lib 에서 확인하며, 하나의 프로젝트 이름(예를 들면 myproject)으로 jar 파일이 생성된다.

6.1.3.예시3) 어플리케이션 빌드 후 WAR 파일 배포

어플리케이션 전체 빌드 후, war 파일을 생성하는 경우에 사용한다.

```
gen build -war
```

빌드된 결과를 프로젝트가 생성된 폴더/dist/ 폴더 하위에서 확인한다. 프로젝트가 생성된 폴더/dist 폴더에는 프로젝트 명으로 된 war 파일이 생성된다.(예)myproject.war)



참고

현재 Anyframe Gen에서는 어플리케이션 빌드 관련된 빌드 스크립트를 [Gen Home]/cli/scripts/application-build.xml 파일에서 관리하고 있다. 이 빌드 파일 내에서 target명이 "build"인 경우가 어플리케이션 빌드에 해당된다.

```
<target name = "build" depends = "clean, dist, war"/>
총략...
```

만약 어플리케이션 빌드 프로세스 내용 중 일부를 변경시키고자 한다면 application-build.xml 파일 내용을 변경하도록 한다.

6.2.프로젝트 빌드

Command Line Interface를 이용해 프로젝트에 대한 빌드를 수행한다. Ant 프로젝트의 경우 build.xml 파일을 이용하여 command를 실행하고, Maven 프로젝트의 경우 pom.xml 파일을 이용하여 빌드를 수행한다.

6.2.1.Ant 프로젝트 빌드

Ant를 이용해 프로젝트를 빌드할 경우

- 1 빌드 대상 프로젝트 폴더 하위로 이동한다. (예를 들어, myproject 라는 프로젝트를 생성시켰다면 myproject 폴더 하위에서 command를 실행시키도록 한다.)
- 2 ant를 실행하기 위한 build.xml, build.properties 파일이 있는지 확인한다.
- 3 다음의 프로젝트 빌드 command를 입력한다.

```
ant [targets]
```

- ex) ant all
- 위 예제는 프로젝트에 대해서 all이라는 target을 실행시키는 ant command로 아래와 같은 target 중 하나를 선택하여 프로젝트 빌드 프로세스를 수행시킬 수 있다. target을 입력하지 않고 ant만 실행시킨 경우 default target인 "default"가 수행된다.

표 6.2. ant targets

target	description
default	프로젝트에 대해서 초기화, 컴파일, 패키징 등의 기본 빌드 프로세스를 수행한다.
deploy	default target을 수행한 후 web type 프로젝트의 경우 war 파일 압축이 풀려진 형태의 폴더를 dist 폴더 하위에 배포한다.
all	위에 설명한 "default" target에서 수행하는 빌드 프로세스 작업 외에 테스트 케이스 수행, 소스 코드 커버리지, CheckStyle/JDepend/FindBugs/Code Duplication 등의 코드 품질 검사 리포팅까지의 작업을 추가로 더 수행한다.
그외	"clean", "init", "resources", "compile", "test-resources", "test-compile", "package", "emma-jars", "test", "report", "deploy" 등 빌드 프로세스 작업 내역 하나 별로 하나의 ant target이 구성되어 있으므로 위에서 설명한 "default"와 "all" target 이외의 작업 별 빌드를 수행해야 할 필요가 있으면 사용하도록 한다.

6.2.2.Maven 프로젝트 빌드

Maven을 이용해 프로젝트를 빌드할 경우

- 1 빌드 대상 프로젝트 폴더 하위로 이동한다. (예를 들어, myproject 라는 프로젝트를 생성시켰다면 myproject 폴더 하위에서 command를 실행시키도록 한다.)
- 2 Maven를 실행하기 위한 pom.xml 파일이 있는지 확인한다.
- 3 다음의 프로젝트 빌드 command를 입력한다.

```
mvn [goals]
```

- ex) mvn package
- 위 예제는 프로젝트에 대해서 package라는 goal을 실행시키는 maven command로 프로젝트에 대한 package 작업을 수행한다. 결과 파일은 프로젝트가 생성된 폴더 하위의 target 폴더에 존재한다.

테스트 케이스 수행, 소스 코드 커버리지, JDepend/Code Duplication 등의 코드 품질 검사 리포팅까지의 작업을 추가로 더 수행하고 싶은 경우, "**site**" 라는 goal을 추가로 더 실행시키면 된다. 리포팅 파일은 프로젝트가 생성된 폴더 하위의 target/site 폴더에 존재한다.

* HSQL DB 이외의 Oracle, MySQL 등 다른 DB를 사용하는 경우, 해당 DB에 대한 driver class를 얻기 위해 pom.xml 파일에 dependency 를 추가해야 한다.

* 예) DB가 Oracle 인 경우

```
<dependency>
  <groupId>ojdbc</groupId>
  <artifactId>ojdbc</artifactId>
  <version>14</version>
</dependency>
```



참고

현재 Anyframe Gen에서는 프로젝트 빌드 스크립트를 [Gen Home]/cli/scripts/project-build.xml 파일에서 관리하고 있다. (project-build.properties 함께 제공) 이 빌드 파일 내 target명들 앞에 prefix로 "shared:"가 붙어있어서 실제 생성한 프로젝트 내 빌드 파일 (build.xml)에서는 이 공통 프로젝트 빌드 파일(project-build.xml)의 target명을 호출하여 실제 프로젝트 빌드를 수행하게 되며, 각 프로젝트 별로 특화된 빌드 프로세스를 추가해 넣을 수도 있다.

만약 공통 프로젝트 빌드 파일을 변경하여 나머지 모든 프로젝트 내 빌드 프로세스 내용을 변경시키고자 한다면 project-build.xml 파일 내용을 변경하도록 한다.

7. 어플리케이션 빌드 및 실행

Command Line Interface를 이용해 어플리케이션 전체 빌드를 수행하고 웹 어플리케이션까지 실행시킬 수 있다. 즉, command를 통해 프로젝트 빌드와 배포를 수행한 후 Jetty를 이용하여 웹 어플리케이션을 구동시킴으로써 어플리케이션의 기능 확인이 가능하다.

7.1. 어플리케이션 빌드 및 웹 어플리케이션 실행

Command Line Interface를 이용해 어플리케이션을 빌드하고 웹 어플리케이션을 실행시켜본다. 이 command는 프로젝트가 web 타입인 경우에만 이용가능하다.

7.1.1. Ant 프로젝트 빌드 및 실행

Ant를 이용해 프로젝트를 빌드할 경우

1. 프로젝트가 생성된 폴더 하위로 이동한다. (예를 들어, myproject 라는 프로젝트를 생성시켰다면 myproject 폴더 하위에서 command를 실행시키도록 한다.)
2. 다음의 어플리케이션 빌드 및 웹 어플리케이션 실행 command를 입력한다.

```
gen run [-options]
```

- 앞에서 설명된 어플리케이션 빌드 과정을 동일하게 수행한 후 웹 어플리케이션을 구동시켜주는 역할을 담당한다.
- ex) gen run
- 위 예제는 추가 옵션 정보 없이 해당 어플리케이션을 빌드하고 Jetty로 실행시키는 command이다.

표 7.1. run options

option	description	default value
-deploy	프로젝트 배포 형태를 정의할 수 있다. "class" 혹은 "jar"를 입력한다. - WEB-INF/classes 폴더 하위에 클래스 형태로 배포할 때 class로 설정 - WEB-INF/lib 폴더 하위에 jar 파일 형태로 배포할 때 jar로 설정	class
-war	-war를 쓰면 프로젝트가 생성된 폴더 하위의 dist 폴더에 프로젝트 명.war 파일을 생성한다.	N/A
-clean	-clean을 쓰면 프로젝트가 생성된 폴더 하위의 dist 폴더 전체를 삭제한 후 빌드를 수행한다.	N/A

3. 웹 어플리케이션 실행 결과를 웹 브라우저를 띄워서 확인하도록 한다. 요청 URL에서 Port 정보가 **8090** 인 것에 유의하도록 한다.

예제의 경우 요청 URL은 `http://localhost:8090/myproject` 가 되며, 이 때, 설치된 플러그인 목록이 조회된다. `http://localhost:8090/myproject/index.jsp` 로 접속하면 Gen 을 통해 생성한 메뉴를 확인할 수 있다.

4. Jetty 폴더를 확인한다.

프로젝트 dist 폴더 하위에 jetty-temp라는 폴더가 생성되어 Jetty가 구동되게 된다. 배포 이후에 클래스나 JAR 파일, JSP 파일 등을 반영하고 Jetty로 재확인하고 싶다면 이 폴더의 해당 위치에 배포하도록 한다.

7.1.2.Maven 프로젝트 실행

Maven을 이용해 프로젝트를 실행 할 경우

1. 프로젝트가 생성된 폴더 하위로 이동한다. (예를 들어, myproject 라는 프로젝트를 생성시켰다면 myproject 폴더 하위에서 command를 실행시키도록 한다.)
2. 다음의 어플리케이션 빌드 및 웹 어플리케이션 실행 command를 입력한다.

```
mvn jetty:run
```

- Maven jetty:run 을 이용해 웹 어플리케이션을 구동시켜준다.
 - ex) mvn jetty:run
 - 위 예제는 어플리케이션을 빌드하고 Jetty로 실행시키는 command이다.
3. 웹 어플리케이션 실행 결과를 웹 브라우저를 띄워서 확인하도록 한다. gen 의 "run" command 를 통해 실행 된 jetty port 와 달리 **8080** port를 사용한다. 예제의 경우 요청 URL은 http://localhost:8080/myproject 가 되며, 이 때, 설치된 플러그인 목록이 조회된다. http://localhost:8080/myproject/index.jsp 로 접속하면 Gen 을 통해 생성한 메뉴를 확인할 수 있다.
 4. target 폴더를 확인한다.

프로젝트 target 폴더 하위에 classes, test-classes, work라는 폴더가 생성되어 Jetty가 구동되게 된다.

IV.Eclipse Plugin

Anyframe Gen Eclipse Plugin을 사용하여 프로젝트 생성, Domain 클래스 및 CRUD 소스 코드 생성, Configuration, JDBC Setting, Anyframe 플러그인 install/uninstall, Hudson 연계 등의 작업을 할 수 있다. 프로젝트는 Wizard를 통해서 생성하며, 그 밖의 작업은 Anyframe Gen Editor를 이용한다.

8.Installing Anyframe Gen Plugin

8.1.Anyframe Gen 플러그인 설치

Anyframe Gen은 압축파일로 배포되며, 설치방법은 [Anyframe Gen 매뉴얼의 설치]를 참고하도록 한다.

1. Anyframe Gen 플러그인 설치는 Anyframe Plugin Update Site를 통해서 설치할 수 있다. 외부 인터넷 연결이 안되거나 네트워크에 문제가 있는 경우 아래의 노트 [Update Site를 이용하지 않고 Anyframe Gen Eclipse Plugin 설치하기] 내용을 참고하여 설치하도록 한다.

ㄱ.Eclipse의 Help > Install New Software... 메뉴를 선택한다.

ㄴ.[Available Software] 화면: 우측 상단의 Add...버튼을 클릭하여 Anyframe Plugin Update Site를 추가하도록 한다. 이때 Name에는 Anyframe Plugin Update Site를, Location에는 http://dev.anyframejava.org/update를 입력한다.

ㄷ.[Available Software] 화면: 중앙에 Anyframe Plugin Update Site를 루트로 하는 Plugin Tree가 나타난다. (Anyframe Common, Anyframe Gen, Anyframe Oden, Anyframe Query Manager 목록 조회됨) 여기서 하단 항목 중 "Contact all update sites during install to find required software" 체크 박스를 해제하여 다른 Plugin이 함께 설치되는 것을 방지한다.

ㄹ.[Available Software] 화면: Plugin Tree 중 Anyframe Gen을 선택하고 하단의 Next 버튼을 클릭한다.

ㅁ.[Install Details] 화면: Anyframe Gen하위로 Anyframe Common plugin이 보이게 된다. Anyframe Gen 사용 시 Anyframe Common이 필요하므로 참조관계에 의해 함께 설치된다. 확인 후 Next 버튼을 클릭한다.

ㅂ.[Review Licenses] 화면: 좌측의 Plugin을 선택하면 우측에 License text가 나온다. Anyframe Common과 Anyframe Gen에 대해서 우측 하단의 "I accept the terms of the license agreements" 구문에 동의하고 Finish 버튼을 클릭한다.

ㅅ.[Security Warning] 화면: 팝업창에서 설치 진행 여부 물을 때 OK 버튼을 선택한다.

ㅇ.[Software Updates] 화면: 팝업창이 뜨고, "Would you like to restart now?" 질문에 Yes 버튼을 선택한다.



Anyframe Plugin Update Site를 이용한 설치 전 주의 사항

Anyframe Gen Plugin이 설치하려고 하는 Eclipse의 dropins 혹은 addins 폴더에 이미 설치되어 있다면 Update Site를 통해 설치될 Plugin과 중복되어 문제를 일으킬 수 있으므로 반드시 dropins 혹은 addins 폴더에 Anyframe Gen Plugin이 설치되어 있는지 확인하고 설치되어 있다면 제거한 후, Update Site를 통해 설치하도록 한다. Update Site를 통해 설치한 경우 물리적인 Anyframe Gen Plugin 파일은 plugins 폴더 하위에 위치하게 된다.

2. "Help > About Eclipse Platform (또는 About Eclipse SDK) > Plug-in Details"를 통해 정상적으로 설치되었는지 확인한다.

☞ About Eclipse Platform (또는 About Eclipse SDK) 메뉴는 Windows용 Eclipse의 경우 Help 메뉴에서, Mac OS X용 Eclipse의 경우 Eclipse 메뉴에서 확인할 수 있다.

☞ Eclipse 루트 폴더 하위의 plugins 폴더 내로 anyframe.common.eclipse.core_xxx.jar와 anyframe.gen.eclipse.core_xxx.jar 파일이 설치되어있는지 확인해볼 수 있다.



3. Anyframe Gen 플러그인을 포함한 전체 Anyframe Gen의 설치가 완료되었다. Anyframe Gen의 기능을 사용하여 개발하기 이전에 반드시 [Preferences] 내용을 참고하여 환경 설정을 모두 끝마치도록 한다.



Update Site를 이용하지 않고 Anyframe Gen Eclipse Plugin 설치하기

Anyframe Gen 플러그인 설치에 plugins 폴더내에 있는 2개의 플러그인 패키지 파일들(anyframe.[제품명].eclipse.*_x.x.x.jar)를 "[eclipse 설치 폴더]/dropins/anyframe/eclipse/plugins" 로 복사시키면 된다. plugins 폴더에는 다음 두가지 플러그인 패키지가 존재한다.

- anyframe.common.eclipse.core_x.x.x.jar : Anyframe plugins의 공용 라이브러리 및 공통 메뉴 제공
- anyframe.gen.eclipse.*_x.x.x.jar : Anyframe Gen 플러그인

☞ 현재 **Anyframe Common** 플러그인의 경우, Anyframe Gen 설치 파일 내에 함께 배포되어 손쉽게 설치할 수 있도록 제공하고 있다. 하지만 Anyframe Common 플러그인은 독립적으로 배포되고 사용될 수 있는 플러그인이므로 추후 Anyframe Common 플러그인 버전이 업데이트되고, 업데이트된 버전을 사용하고자 한다면 Anyframe Gen에서 배포한 구 버전 Anyframe Common 플러그인을 제거하고 업데이트된 플러그인 패키지를 설치하도록 한다.

8.2.Hudson 연계 플러그인 설정하기

Anyframe Gen Editor에서 Hudson 과 연계하여 Hudson Job에 대해 추가,수정,삭제,실행 등의 작업을 하기 위해서는 다음과 같은 설정 작업이 필요하다.

1. Anyframe-gen-X.X.X-bin/plugins/hudson 폴더에 있는 플러그인 패키지 파일 (**anyframe.gen.eclipse.hudson-X.X.X.jar**)를 "[Hudson 설치 폴더]/hudson/war/WEB-INF/lib" 로 복사

anyframe.gen.eclipse.hudson-X.X.X.jar 은 Gen에서 설정한 정보를 Hudson에 반영하는 역할을 수행한다.

2. [Hudson 설치 폴더]/hudson/war/WEB-INF/web.xml 파일을 열어서 아래와 같은 내용을 추가한다. hudsonHome, hudsonJobDir 는 [Hudson 설치 폴더]/bin 기준으로 표현된 것으로, 경로가 다를 경우 수정이 필요하다. hudsonHome의 경우 hudson.tasks.Ant.xml, hudson.tasks.Mailer.xml, hudson.tasks.Maven.xml 세개 파일이 있는 경로를 지정한다. hudsonJobDir의 경우 hudson job 이 있는 폴더경로를 지정한다.

```
<servlet>
  <servlet-name>Hudson Gen Servlet</servlet-name>
  <servlet-class>anyframe.gen.eclipse.hudson.HudsonGenServlet</servlet-class>
  <init-param>
```

```
<param-name>hudsonHome</param-name>
<!-- 상대경로는 [Hudson 설치 폴더]/bin 기준으로 표현된 것 -->
<!-- Hudson을 start 하는 경로가 위의 경로가 아닌 경우 param-value 조정해야 함 -->
<param-value>..\hudson</param-value>
</init-param>
<init-param>
<!-- 상대경로는 [Hudson 설치 폴더]/bin 기준으로 표현된 것 -->
<!-- Hudson을 start 하는 경로가 위의 경로가 아닌 경우 param-value 조정해야 함 -->
<param-name>hudsonJobDir</param-name>
<param-value>..\hudson\jobs</param-value>
</init-param>
</servlet>
<servlet-mapping>
<servlet-name>Hudson Gen Servlet</servlet-name>
<url-pattern>/anyframe/api/*</url-pattern>
</servlet-mapping>
```

9. Preferences

9.1. Eclipse 설정

Anyframe Gen 을 이용하여 생성한 프로젝트 실행을 위해 필요한 환경설정을 한다. Anyframe Gen의 환경설정은 Eclipse Preferences 창을 통해 수행한다.

- ClassPath 설정 : Anyframe Library 를 인식하도록 ClassPath를 설정한다.

Window >> Preferences >> Java >> Build Path >> Classpath Variables 에서 New 클릭

Name : GEN_REPO

Path : [Gen Home]\repo [Folder..]

- Server 설정 : 어플리케이션이 구동될 WAS로 Tomcat v6.0을 사용한다.

Window >> Preferences >> Server >> Runtime Environments 에서 Add..클릭

Name : Apache Tomcat v6.0

Tomcat Installation directory : [Tomcat 이 설치된 root 경로]

JRE : Workbench default JRE

- Ant Home 설정 : 설치된 Ant 를 인식하도록 Ant Home을 변경하거나 Ant Home Entries에 jdepend-X_X.jar파일을 추가한다.

- Ant Home 변경 : **Window >> Preferences >> Ant >> Runtime >> Classpath Tab >> 우측 하단의 Ant Home..** 버튼 클릭

path : [Gen Home]\ant

- Ant Home Entries 파일 추가 : **Window >> Preferences >> Ant >> Runtime >> Classpath Tab >> 우측 의 Add External JARs...** 버튼 클릭

path : [Gen Home]\ant\lib\jdepend-2.9.jar 파일 선택

환경설정을 통해 Anyframe Gen으로 생성된 프로젝트가 정상적으로 실행이 된다.

9.2. DB 구동

- 샘플 DB를 구동 시킨다.

현재 Anyframe Gen은 샘플 DB로 HSQL DB를 제공하고 있다. [Gen Home]\db\scripts\hsqldb 하위의 start.cmd 파일을 실행시켜서 DB를 구동한다.

탐색기로 해당경로 이동 후, start.cmd 파일을 더블클릭하면 hsqldb가 구동된다. 만약 Oracle과 같은 다른 DB를 사용한다면 DB를 구동시킨 후, [Gen Home]\db\scripts\[DB명]\sampledb-[DB명].sql 파일의 스크립트를 실행해준다. 현재는 Oracle(sampledb-oracle.sql), MySQL(sampledb-mysql.sql), Sybase(sampledb-sybase.sql)에 대한 샘플쿼리문이 제공되고 있으며, Anyframe Gen 에서 자동으로 설치되는 샘플이 정상적으로 동작하기 위해 필요한 것이다.

구동된 DB에 대한 JDBC 설정은 프로젝트를 생성하면서 반영하거나, Anyframe Gen JDBC Setting 탭을 통해서 반영이 가능하다. 자세한 내용은 [New Project Creation]또는 [JDBC Setting]을 참고하도록 한다. JDBC Setting 탭은 Domian Type 프로젝트 생성 후에 사용이 가능하다.


9.3.MAVEN_HOME 설정

- Anyframe Gen 1.5.0 버전부터 Maven 프로젝트 생성 기능이 추가되었다.

Maven 프로젝트를 정상적으로 생성시키기 위해서는 로컬에 Maven이 설치되어 있어야 한다. Maven 설치 후 시스템 변수 MAVEN_HOME 을 추가하고 속성값을 Maven 설치 폴더로 지정해준다. 또한, 시스템 변수 PATH에 %MAVEN_HOME%\bin을 추가한다.

10. New Project Creation

Anyframe Gen 을 이용해 프로젝트를 생성 할 수 있다.

- [Create new project] : 프로젝트 생성
- [프로젝트 생성 시 발생할 수 있는 에러 해결 Tip]
 - [Out of memory] : 프로젝트 생성 시 java.lang.OutOfMemoryError 조치방법
 - [Project Clean & JSP Validation Error]: 프로젝트 생성 시  표시가 뜰 경우 조치방법

10.1. Create new project

1. File >> New >> Other.. >> Anyframe >> Project 을 선택
2. 열린 창에서 프로젝트 생성을 위해 다음과 같은 내용을 입력한다. 아래 항목을 문제없이 기입하는 경우 Next 버튼과 Finish 버튼이 활성화 되며, Next 버튼을 클릭하면 DB정보를 구성할 수 있는 화면으로 연결된다.
 - Project Name: 프로젝트 이름으로, 예제의 경우 emarketplace로 설정하였다.
 - Create new ant project : Ant 프로젝트 생성하기 위해 선택한다. build.xml 이 생성된다.
 - Create new maven project : Maven 프로젝트 생성하기 위해 선택한다. pom.xml 이 생성된다. 아래 3개 항목이 활성화 된다.
 - Group Id : Group ID
 - Artifact Id : Artifact ID
 - Version : Version
 - Gen Home: Anyframe Gen을 설치하기 위해 설치 시 선택했던 경로
 - Base Location: 프로젝트 생성 위치로, default로 지정되는 경로를 꼭 사용할 필요 없이 원하는 경로로 변경 가능하다.
 - Package Name: 패키지 이름으로, 예제의 경우 com.sds.emp로 지정하였다.
 - Web: Web 타입 프로젝트 생성하는 경우 선택한다. Dynamic Web 프로젝트가 생성되며 Anyframe foundation 플러그인이 샘플과 함께 설치된다.
 - Service: Service 타입 프로젝트 생성하는 경우 선택한다. Java 프로젝트가 생성되며 Anyframe foundation 플러그인 라이브러리만 설치되고 샘플은 설치되지 않는다. annotation 기반으로 자바 코드를 생성할 수 있는 Anyframe Gen 아키텍처가 설치된다.
3. DB정보를 구성하기 위해 위 화면에서 Next 버튼을 클릭하였다면 다음과 같은 JDBC Configuration 화면이 제공된다.

Finish 버튼을 누르면 프로젝트가 생성된다.

- Database Type: Database 타입으로, HSQL, Oracle, MySQL, Sybase를 제공하고 있다.
- Database Name: Database 의 이름
- Schema: 스키마의 이름
- User Name: DB 유저 이름

- Password: 패스워드
 - Server: DB가 설치된 서버 ip 정보, 로컬에서 사용하는 경우 localhost
 - Port: HSQL의 경우 -1, Oracle의 경우 1521, MySQL의 경우 3306, Sybase는 3000
 - Hibernate Dialect: Hibernate에서 쿼리 수행 시, DBMS에 최적화된 기능을 제공하기 위해 사용되는 것이 SQL Dialect 이며, 이 Dialect 프로퍼티를 사용하여 해당 DB 별 Dialect 정보를 설정할 수 있다. 각 DB 별 Dialect 클래스가 따로 존재하여 HSQL DB를 선택한 경우, 디폴트로 org.hibernate.dialect.HSQLDialect값이 선택된다.(Oracle, MySQL, Sybase Dialect 도 제공함)
 - Driver Class Name: DB의 드라이버 클래스를 설정해 주기 위한 값으로, HSQL의 경우 org.hsqldb.jdbcDriver 값으로 설정된다. Database Type 선택 시 해당 DB에 맞는 값이 셋팅된다.
 - Driver Jar Path : Anyframe Gen 설치 시 샘플 DB를 위한 HSQL DB Driver Jar 파일이 제공되므로 디폴트로 설정된다. 만약 다른 DB를 사용한다면 해당 DB의 Driver jar 파일로 연결시켜준다.
4. 프로젝트 생성이 완료되면 Eclipse 내에 프로젝트가 정상적으로 생성되어 Import 되었는지 확인해 본다. Ant 프로젝트의 경우 build.xml 빌드스크립트가 존재하고, Maven 프로젝트의 경우 pom.xml 파일이 존재한다. Maven 프로젝트의 경우 프로젝트가 라이브러리를 가지고 있지 않기 때문에, 생성된 Maven 프로젝트에 대해서 M2Eclipse 를 Enable Dependency Management 해 줘야 한다. (**Maven 프로젝트 선택 >> Maven >> Enable Dependency Management**) Web 타입 프로젝트일 경우 dynamic 웹 프로젝트로 생성되고, Service 타입 프로젝트일 경우 자바 프로젝트로 생성된다.
5. 서버를 실행하여 프로젝트를 실행해본다.
- Web 타입 프로젝트일 경우, 서버를 실행하여 프로젝트를 실행해본다. 프로젝트를 선택한 후 마우스 우측 버튼 클릭 >> Run As >> Run On Server 메뉴를 선택하고 이때 기존에 설정되어있던 **Tomcat Server(6.0 Version 사용)**가 없는 경우 등록해주고 실행하도록 한다.
- Web 타입 프로젝트의 경우 서버를 start 했을 때 열리는 웹 페이지는 설치된 플러그인 목록이 보이는 화면이다. 프로젝트 생성 이후에 Anyframe Gen Editor의 CRUD Generation 기능을 통해 생성되는 화면은 /src/main/webapp/index.jsp를 선택한 후 마우스 우측 버튼 클릭하여 Run As 함으로써 확인할 수 있다.
- 프로젝트를 생성을 통해 만들어진 프로젝트 설정 파일(Properties)을 수정하여, 수정된 설정 값을 통해 소스가 생성되도록 할 수 있다. 아래 표에 설명된 항목별 내용을 보고 DB 설정을 비롯하여 다양한 공통 속성 값을 변경시킬 수 있다. 현재 DB 속성 정보들의 디폴트 값은 HSQL 샘플 DB에 대한 속성 값으로 설정되어 있다.

• 표 10.1. build.properties I (프로젝트 정보)

Property Name	Description	Required	Default Value
gen.home	Anyframe Gen을 설치한 루트 폴더	Y	[Gen Home]
project.build	프로젝트 빌드도구	Y	ant
project.name	프로젝트 명	Y	myproject
project.version	프로젝트 버전	Y	1.0.0
project.home	프로젝트 루트 폴더 경로	Y	[Gen Home] \ applications \ [프로젝트명]
package.name	프로젝트 대표 패키지 명	Y	프로젝트명
dao.framework	DAO Framework 선택(query, hibernate 중 택일)	Y	query
emma.enabled	프로젝트 빌드 시 emma를 이용하여 코드 커버리지 리포트 결과 생성 여부	Y	true
template_type	프로젝트 및 CRUD 소스 코드 생성 시 기반이 되는 Template 형태(ex. default, miplatform 등)	Y	default
project.type	프로젝트 타입	Y	web
web.context.path	프로젝트 빌드도구가 ant일 경우에만 의미있는 값으로, jetty를 이용하여 웹 어플리케이션을 실행할 때 사용되는 WebContext Path명	Y	프로젝트명

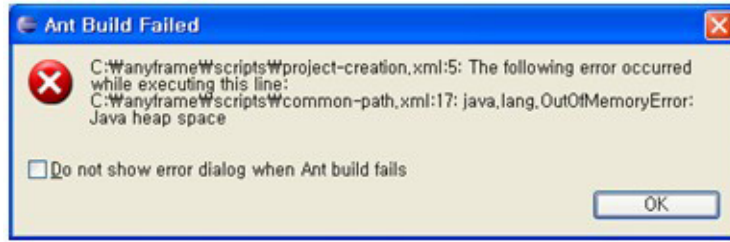
표 10.2. build.properties II (DB 정보)

Property Name	Description	Required	Default Value
db_name	[DB] Database 명	Y	sampledb
dialect	[DB] DB별 Hibernate Dialect 클래스 설정	Y	org.hibernate.dialect.HSQLDialect
driver_class	[DB] DB의 드라이버 클래스 설정	Y	org.hsqldb.jdbcDriver
schema	[DB] schema 명	Y	PUBLIC
username	[DB] DB 사용자 명	Y	sa
password	[DB] DB 사용자 패스워드	N	N/A
url	[DB] DB에 접근하기 위한 Access URL	Y	jdbc:hsqldb:hsq://localhost/sampledb
lobhandler	[DB]BLOB,CLOB과 같은 Large Data 타입에 대한 핸들링	N	N/A
port	[DB] DB에 접근하기 위한 port	Y	-1
driver_jar_path	[DB] DB에 접근하기 위한 driver jar파일의 경로 이때, 패스 표현 방법에 주의를 기울여야 함 상대경로로 입력시 경로 처음에 '/'로 시작하게 될 경우 절대경로로 인식하게 됨	Y	C:\\Gen-1.5.1\\db\\lib\\hsqldb-1.8.0.10.jar
server	[DB] DB에 접근하기 위한 server정보	Y	localhost
db_type	[DB] DB 정보(HSQL, Oracle, MySQL, Sybase)	Y	HSQL
paging_generator	[Query Service에서 사용] 목록 조회 시 페이징 처리를 지원하기 위한 SQL 생성 클래스명을 설정 ex. Oracle: anyframe.core.query.impl.jdbc.generator.OraclePagingSQLGenerator ex. MySQL:anyframe.core.query.impl.jdbc.generator.MySQLPagingSQLGenerator ex. Sybase:anyframe.core.query.impl.jdbc.generator.DefaultPagingSQLGenerator	Y	anyframe.core.query.impl.jdbc.generator.HSQLPagingSQLGenerator

10.2.프로젝트 생성 시 발생할 수 있는 에러 해결 Tip

10.2.1.Out of memory

프로젝트 생성 시 다음과 같은 out of memory 에러가 발생할 경우 다음과 같이 조치한다.



1. 실행하고 있는 eclipse가 설치되어 있는 폴더로 이동한다.
2. root 폴더 하위의 eclipse.ini 파일의 내용에서 launcher.XXMaxPermSize값(ex. 256M)을 높여서 재설정해준다. 필요 시 메모리 사용 최소값(-Xms)과 최대값(-Xmx)을 증가시켜서 사용하도록 한다. 다음은 Eclipse 3.5.0(Galileo) 버전의 eclipse.ini 파일 예이다.

```
-clean
-startup
plugins/org.eclipse.equinox.launcher_1.0.201.R35x_v20090715.jar
--launcher.library
plugins/org.eclipse.equinox.launcher.win32.win32.x86_1.0.200.v20090519
-product
org.eclipse.epp.package.jee.product
--launcher.XXMaxPermSize
256M
-showsplash
org.eclipse.platform
--launcher.XXMaxPermSize
128m
-vmargs
-Dosgi.requiredJavaVersion=1.5
-Xms400m
-Xmx512m
```

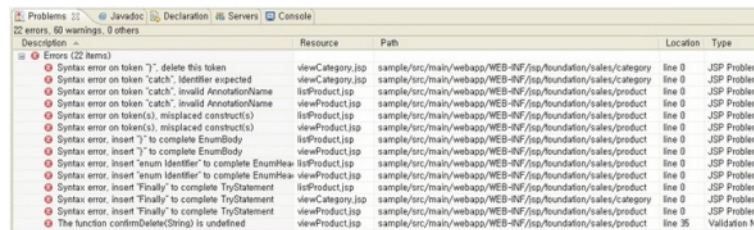
10.2.2.Project Clean & JSP Validation Error

- 프로젝트 생성 시 다음과 같이 생성된 프로젝트에 🗑️ 표시가 뜨는 경우, 프로젝트를 clean을 하거나 해당 프로젝트를 close 한 후 다시 open하면 표시가 사라진다.

☞ Project >> Clean... 수행

☞ 프로젝트 선택 후 오른쪽 마우스 클릭 >> Close Project 후 프로젝트 선택 후 오른쪽 마우스 클릭 >> Open Project

- Problems 에 JSP Validation 에러가 나오는 경우, 실제로 error는 아니지만 eclipse에서 인식을 에러로 하는 것으로 다음과 같이 validation 설정을 변경해주면 된다. Eclipse 내 프로젝트에서 Validation 기능을 사용하게 되면 시간이 많이 소요될 수 있다. 반드시 필요한 경우가 아니고, 속도 향상을 위해서라면 Validation 설정을 모두 Disable All 버튼을 선택하여 사용하지 않는 것으로 설정할 수도 있다.



☞ Window >> Preferences... >> Validation >> JPA Validator, JSP Syntax Validator 의 Build 항목 체크 해제 후 Apply

11.Domain Generation

Anyframe Gen을 이용해 Domain Class 를 생성해본다.

DB 테이블 기반으로 Domain Class를 생성시키기 때문에 JDBC 설정이 제대로 되었는지 [JDBC Configuration 설정방법]을 참조하여 확인 한 후, Domain Class를 생성하도록 한다.

- 1 프로젝트를 선택한 후 우 클릭 >> **Anyframe Gen** 메뉴를 선택한다. 이때 해당 Service 프로젝트가 Domain 타입 프로젝트가 아닌 경우, Anyframe Gen 메뉴는 비활성화되어 보여진다.
- 2 Anyframe Gen Editor에서 Domain Generation Tab을 선택한다. Refresh 버튼을 수행하면 현재 선택 가능한 DB Table들과 소스 코드 패키지가 트리 형태로 조회된다. 이때 Domain Class로 생성하고자 하는 Table을 선택하고, Domain Class를 어느 소스 코드 패키지 하위로 생성시킬 것인지를 선택한 후 Generate 버튼을 클릭한다.

참조관계에 있는 테이블의 경우 해당 테이블을 동시에 선택하여 Domain Class를 생성 해야 하며, 이를 통해 참조관계에 관한 정보가 제공된다.

- Choose Tables: JDBC 구성에 따라 연결된 DB의 테이블 목록이 조회된다.
 - Choose a package: 해당 프로젝트에 있는 소스 코드 패키지 경로가 조회된다.(src/main/java 소스 폴더에 한함)
- 3 Domain Class 가 제대로 생성되었는지 확인해 본다. 원하는 패키지 하위로 Domain Class들이 위치하고 있는지 확인한다.

생성된 Domain Class는 JPA Annotation 설정을 통해 DB Table, Column 정보들을 가지고 있다.

```
@Entity
@Table(name = "BOARD", schema = "PUBLIC")
public class Board implements Serializable {
    private BoardId id;
    private BoardMaster boardMaster;
    private String boardName;
    중략...
    @EmbeddedId
    @AttributeOverrides({@AttributeOverride(name = "boardId", column =
        @Column(name = "BOARD_ID", nullable = false)
    )
    , @AttributeOverride(name = "boardMasterId", column =
        @Column(name = "BOARD_MASTER_ID", nullable = false)
    )
    })
    public BoardId getId() {
        return this.id;
    }
    public void setId(BoardId id) {
        this.id = id;
    }
    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "BOARD_MASTER_ID", nullable = false,
        insertable = false, updatable = false)
    public BoardMaster getBoardMaster() {
        return this.boardMaster;
    }
    public void setBoardMaster(BoardMaster boardMaster) {
        this.boardMaster = boardMaster;
    }
    @Column(name = "BOARD_NAME", nullable = false, length = 150)
    public String getBoardName() {
```

```
return this.boardName;
}
public void setBoardName(String boardName) {
    this.boardName = boardName;
}
}
```



참고

위에서 설명한 기능 수행 도중에는 프로젝트 생성, 코드 생성 및 DB 변경 등과 같은 Anyframe Gen의 기능을 동시에 수행시킬 수 없으므로 유의하도록 한다. (Background 실행을 통한 기능 수행 시)

12.CRUD Generation

Anyframe Gen을 이용해 CRUD를 생성해본다.

Domain Class를 중심으로 CRUD에 대한 기본 코드를 생성하고 자동으로 테스트 코드와 테스트 데이터를 생성해줌으로써 생성된 코드의 기능 확인까지 손쉽게 할 수 있다. 따라서, CRUD Generation 기능을 사용하기 전에 Domain Generation 기능을 이용하여, 연결된 DB 테이블 기반의 Domain Class를 먼저 생성하도록 한다.

- 1 프로젝트를 선택한 후 우 클릭 >> **Anyframe Gen** 을 선택한다.
- 2 Anyframe Gen Editor에서 CRUD Generation Tab을 선택한다. Refresh 버튼을 수행하면 현재 선택 가능한 Domain Class들과 타겟 프로젝트로 지정된 프로젝트 목록이 조회된다.
- 3 CRUD 기능을 구현하고 싶은 Domain Class를 선택하고, CRUD Project Configuration 내용을 입력한다.
 - CRUD Service: CRUD 기능이 구현될 서비스 명으로 Default로 이름이 지정되며 조회만 가능하다.
 - Package: CRUD가 생성될 패키지 이름이며, 특정 서브 시스템 하위로 코드를 생성하고 싶다면 중간 패키지 정보를 함께 입력하면 된다. (ex. sub.categories)
 - Project Name: 프로젝트명으로 조회만 가능하다.
 - Web source code generation: 비즈니스 레이어에 해당하는 소스 뿐 아니라 프리젠테이션 레이어 소스 코드들도 함께 생성하고자 한다면 체크 박스에 체크하도록 한다. Service 타입 프로젝트의 경우, 이 체크박스는 활성화 되지 않는다.
- 4 Generate 버튼을 클릭한 후, CRUD 코드가 정상적으로 생성되었는지 확인해본다. 타겟 프로젝트로 지정한 프로젝트에는 Domain Class에 대한 비즈니스 서비스 인터페이스, 구현 클래스, DAO 클래스, Spring 설정 파일들, 테스트 코드, 테스트 데이터 등이 생성된다. Web 타입 프로젝트에는 Spring MVC Controller 클래스, Spring MVC 설정 파일, 테스트 코드, JSP 페이지 등이 생성된다. 참고로, CRUD 생성 후 소스파일에 🚩 표시가 뜨는 경우는, test 플러그인이 없기 때문이다. Anyframe Gen Editor의 Installation 탭에서 test 플러그인을 다운받아 설치하면 해결된다.

- 서비스인터페이스 (src/main/java/[대표패키지]/[-package 값]/service/[Entity 클래스명]Service.java)

```
public interface BoardService{
    void create(Board board) throws Exception;
    void remove(BoardId id) throws Exception;
    중략...
}
```

- 서비스구현클래스 (src/main/java/[대표패키지]/[-package 값]/service/impl/[Entity 클래스명]ServiceImpl.java)

```
@Service("boardService")
@Transactional(rollbackFor = {Exception.class}, propagation = Propagation.REQUIRED)
public class BoardServiceImpl implements BoardService {

    @Inject
    @Named("boardDao")
    private BoardDao boardDao;

    public void create(Board board) throws Exception{
        this.boardDao.create(board);
    }

    public void remove(BoardId id) throws Exception {
```

```

        this.boardDao.remove(id);
    }
    중략...
}

```

- **DAO클래스** (src/main/java/[대표패키지]/[-package 값]/service/impl/[Entity 클래스명]Dao.java)

```

@Repository("boardDao")
public class BoardDaoQueryImpl extends AbstractDAO {

    @Value("#{contextProperties['pageSize'] ?: 10}")
    int pageSize;

    @Value("#{contextProperties['pageUnit'] ?: 10}")
    int pageUnit;

    @Inject
    public void setQueryService(IQueryService queryService) {
        super.setQueryService(queryService);
    }

    /** {@inheritDoc} */
    public void create(Board board) throws Exception {
        super.create("Board", board);
    }

    /** {@inheritDoc} */
    public void remove(BoardId id) throws Exception {
        Board board = new Board();
        board.setId(id);
        super.remove("Board", board);
    }
    중략...
}

```

- **매핑쿼리문** (src/main/resources/sql/mapping-query-[Entity 클래스명].xml)

```

<queryservice>
<queries>
<query id="createBoard">
<statement>
INSERT INTO BOARD (BOARD_ID, BOARD_DESC, BOARD_MASTER_ID, BOARD_NAME,
BOARD_ORDER, BOARD_TOPICS, REG_DATE)
VALUES (:vo.id.boardId, :vo.boardDesc, :vo.id.boardMasterId, :vo.boardName,
:vo.boardOrder, :vo.boardTopics, :vo.regDate)
</statement>
</query>
중략...
</queries>
</queryservice>

```

- **MessageSource** (src/main/resources/message/message-generation.properties)

```

# -- Board-START
board.id.boardId=Board Id
board.id.boardMasterId=Board Master Id
board.boardDesc=Board Desc
board.boardName=Board Name
board.boardOrder=Board Order
board.boardTopics=Board Topics
board.regDate=Reg Date

```

```
# -- success messages --
success.board.create=Board has been added successfully.
success.board.update=Board has been updated successfully.
success.board.delete=Board has been deleted successfully.

# -- error messages --
error.boardserviceimpl.create=Board data not created
error.boardserviceimpl.create.solution=Enter correct data for mandatory field
or enter data according to formats means date format as yyyy-mm-dd
error.boardserviceimpl.create.reason=Entered incorrect data for Board
중략...
```

- 통합테스트케이스 (src/test/java/[대표패키지]/[-package 값]/service/[Entity 클래스명]ServiceTest.java)

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(locations = { "file:./src/main/resources/spring/context-*.xml" })
public class BoardServiceTest{

    @Inject
    @Named("boardDao")
    private BoardDao boardDao;

    @Test
    @Rollback(value=true)
    public void manageBoard() throws Exception {
        // 1. create a new board
        Board board = getBoard();

        if(boardService.get(board.getId()) != null)
            boardService.remove(board.getId());

        boardService.create(board);
        중략...
    }
    중략...
}
```

- 컨트롤러 (src/main/java/[대표패키지]/[-package 값]/web/[Entity 클래스명]Controller.java)

```
@Controller
@RequestMapping("/board.do")
public class BoardController {

    /**
     * Resource Injection on BoardService
     */
    @Inject
    @Named("boardService")
    private BoardService boardService;

    중략...

    /**
     * Display form for adding Board.
     * @param model containing control data
     * @return the prepared form view
     * @throws Exception in case of an invalid new form object
     */
    @RequestMapping(params = "method=createView")
    public String createView(Model model) throws Exception
```

```

{
    model.addAttribute(new Board());

    return "genViewBoard";
}
}

```

- **UI JSP 파일** (src/main/webapp/WEB-INF/jsp/[Entity 클래스 명]/*.jsp)

```

<%@ include file="/sample/common/taglibs.jsp"%>
<html>
<head>
    <%@ include file="/sample/common/meta.jsp" %>
    <title><fmt:message key="boardList.title"/></title>
    <meta name="heading" content="<spring:message code='boardList.heading' />" />
    <link rel="stylesheet" href="<c:url value='/sample/css/admin.css' />"
        type="text/css">

    <script type="text/javascript"
        src="<c:url value='/sample/javascript/CommonScript.js' />"></script>
    <script language="JavaScript">
        function fncCreateBoardView() {
            document.location.href="<c:url value='/board.do?method=createView' />";
        }
    </script>
</head>
</html>

```

- **단위테스트케이스** (src/test/java/[대표패키지]/[package 값]/web/[Entity 클래스 명]ControllerTest.java)

```

@RunWith(JMock.class)
public class BoardControllerTest {

    private BoardController controller;
    private String SUCCESS_CREATEVIEW = "genViewBoard";
    private String SUCCESS_CREATE = "redirect:/board.do?method=list";
    private String SUCCESS_GET = "genViewBoard";
    private String SUCCESS_UPDATE = "redirect:/board.do?method=list";
    private String SUCCESS_LIST = "genListBoard";
    private String SUCCESS_REMOVE = "redirect:/board.do?method=list";
    private Mockery context = new JUnit4Mockery();
    private BoardService mockService = null;

    @Before
    public void setUp() throws Exception {
        System.setProperty("log4j.configuration", "log4j-test.xml");

        this.mockService = context.mock(BoardService.class);
        this.controller = new BoardController();
        this.controller.setBoardService(this.mockService);
    }

    public void setBoardController(BoardController controller) {
        this.controller = controller;
    }

    @Test
    public void testCreateView() throws Exception{
        String viewName = this.controller.createView(new ExtendedModelMap());

        assertEquals("returned correct view name", SUCCESS_CREATEVIEW, viewName);
    }
}

```

종료...



참고

위에서 설명한 기능 수행 도중에는 프로젝트 생성, 코드 생성 및 DB 변경 등과 같은 Anyframe Gen의 기능을 동시에 수행시킬 수 없으므로 유의하도록 한다. (Background 실행을 통한 기능 수행 시)

13. Configuration

Anyframe Gen을 사용하기 위해 Project Location 확인 및 DAO Frameworks, Template 정보를 설정한다.

프로젝트는 기본적으로 Query Service DAO Framework 을 사용하도록 설정되어 있으며, Hibernate/JPA 으로 DAO Frameworks 설정을 변경하고 싶을 경우 Installation Tab에서 Hibernate 플러그인을 설치해야 한다. Hibernate/JPA 플러그인이 설치되어 있지만 예외적으로 Template이 MiPlatform으로 되어 있는 경우는 Hibernate/JPA Framework을 사용할 수 없다.

Template은 기본적으로 default만 제공되고, Installation Tab에서 MiPlatform 플러그인을 설치한 경우 추가적으로 선택할 수 있다.

- 1 프로젝트를 선택한 후 우 클릭 >> **Anyframe Gen** 을 선택한다.
- 2 Anyframe Gen Editor에서 Configuration Tab을 선택하고 프로젝트 경로 확인 및 DAO Framework, Template 정보를 구성한다.
 - Project Location: 현재 프로젝트가 생성된 위치정보가 보인다.
 - DAO Frameworks: Query Service, Hibernate/JPA 중에 하나를 선택할 수 있고, Hibernate/JPA는 Hibernate 플러그인을 설치한 이후에 선택이 가능하다. map, miplatform Template의 경우 Query Service만 지원된다.
 - Template Selection: 기본적으로 default, map 템플릿이 지원되며, default 템플릿은 Annotation 기반의 Spring/SpringMVC 등으로 구성된 소스 코드 템플릿이 제공된다. MiPlatform 플러그인을 설치한 경우 MiPlatform Template도 선택할 수 있다. MiPlatform 의 경우 PID(Presentation Interface Developer)가 설치되어 있어야 생성된 소스가 정상적으로 동작한다.
- 3 입력정보 작성이 끝나면 반드시 Apply 버튼을 클릭하여 Configuration 정보가 저장되도록 한다. 구성된 정보가 정상적으로 반영되었는지 확인하기 위해서는 탐색기에서 선택한 프로젝트 경로로 이동하여, build.properties 파일내의 dao.framework(DAO Frameworks), template_type(Template Type) 항목의 값을 확인해본다. 정상적으로 반영이 되었다면, 해당 항목의 값이 입력한 값들로 변경되어 보일 것이다. 참고로, project.home 은 프로젝트 경로 정보이다.

14.JDBC Setting

Anyframe Gen의 Domain Class 를 생성 기능을 사용하기 위해서 DB 연결정보를 설정한다.

1 프로젝트를 선택한 후 우 클릭 >> **Anyframe Gen** 을 선택한다.

2 Anyframe Gen Editor에서 JDBC Setting Tab을 선택하고 JDBC 연결정보를 입력한다.

- Database Type: Database 타입으로 HSQL, Oracle, MySQL, Sybase 를 제공하고 있다.
- Database Name: Database의 이름
- User Name: DB 유저 이름
- Schema: DB 스키마의 이름
- Password: 패스워드
- Server: DB가 설치된 서버 ip 정보, 로컬에서 사용하는 경우 localhost
- Port: HSQL의 경우 -1, Oracle의 경우 1521, MySQL의 경우 3306, Sybase는 3000
- Hibernate Dialect: Hibernate에서 쿼리 수행 시, DBMS에 최적화된 기능을 제공하기 위해 사용되는 것이 SQL Dialect 이며, 이 Dialect 프로퍼티를 사용하여 해당 DB 별 Dialect 정보를 설정할 수 있다. 각 DB 별 Dialect 클래스가 따로 존재하여 HSQL DB를 선택한 경우, 디폴트로 org.hibernate.dialect.HSQLDialect값이 선택된다.(Oracle, MySQL, Sybase Dialect 도 제공함)
- Driver Class Name: DB의 드라이버 클래스를 설정해 주기 위한 값으로, HSQL의 경우 org.hsqldb.jdbcDriver 값으로 설정된다. Database Type 선택 시 해당 DB에 맞는 값이 셋팅된다.
- Driver Jar Path: Anyframe Gen 설치 시 샘플 DB를 위한 HSQL DB Driver Jar 파일이 제공되므로 디폴트로 설정된다. 만약 다른 DB를 사용한다면 해당 DB의 Driver jar 파일로 연결시켜준다.

3 입력정보 작성이 끝나면 반드시 Apply 버튼을 클릭하여 JDBC Configuration 정보가 저장되도록 한다.

Schema 정보를 리스트에서 선택하지 않고 Apply 버튼을 클릭할 경우, Schema 를 선택하라는 메시지가 출력된다. Schema 를 제외한 나머지 JDBC Configuration 정보가 정상적인 경우 Schema 정보를 얻을 수 있으며, 얻어온 리스트에서 Schema 를 선택하여 Apply 하면 Configuration 정보가 반영된다.

JDBC 환경 설정 정보가 정상적으로 구성되어 DB와 잘 연결되는지 확인하기 위해서는 Domain Generation Tab으로 이동하여 Refresh 버튼을 클릭해본다. 이때 정상적으로 연결이 잘 되는 경우라면 해당 DB의 Table들이 조회될 것이다.



참고

위에서 설명한 기능 수행 도중에는 프로젝트 생성, 코드 생성 및 DB 변경 등과 같은 Anyframe Gen의 기능을 동시에 수행시킬 수 없으므로 유의하도록 한다. (Background 실행을 통한 기능 수행 시)

15.Anyframe Plugin Installation

Anyframe Gen에서 Anyframe에서 제공하는 플러그인 을 Install/Uninstall 하기 위해서 사용한다. 플러그인을 설치하면 해당 플러그인과 연관된 라이브러리가 설치되며, 일부 플러그인의 경우 샘플코드를 제공하고 있다. 제공되는 샘플코드를 라이브러리와 함께 설치하고 싶은 경우, "Install or uninstall plugins with sample"에 체크한다.

1 프로젝트를 선택한 후 우 클릭 >> **Anyframe Gen** 을 선택한다.

2 Anyframe Gen Editor에서 Installation Tab을 선택하면 다음과 같은 항목이 보인다.

- Update plugin list: Anyframe Repository로 부터 Anyframe 플러그인 목록 파일을 update 받는 기능이 수행됨. default로 update 된 파일은 settings.xml 파일이 있는 [GEN HOME]/repo 에 존재하게 됨
- !: Install/Uninstall 할 대상을 체크하는 부분으로, foundation 플러그인은 Anyframe 사용을 위한 기본 플러그인이므로, Install/Uninstall 대상에서 제외된다.
- Plugin Type: 플러그인 타입
- Group Id: 플러그인 Library의 Group 명
- Artifact Id: 플러그인 Library 명
- Version: 플러그인 버전
- Sample: 샘플코드 존재여부로 샘플코드가 있으면 O, 라이브러리만 존재하는 플러그인은 X 로 표시
- Installed: 플러그인이 설치되었는지 여부로 플러그인을 설치하면 O로 표시
- Install or uninstall plugins with sample: 샘플코드를 라이브러리와 함께 설치/삭제하고 싶은 경우 체크박스에 체크하며,Service 타입 프로젝트에서는 비활성화 된다.

3 설치 또는 제거할 플러그인 항목을 체크한 후, Install 혹은 Uninstall 버튼을 클릭한다.

Install/Uninstall 할 경우 tomcat은 stop 시킨 상태에서 수행하도록 한다. 그렇지 않은경우, 동작은 정상적으로 수행되나, 아래 4번 내용과 같은 예러가 발생하게 된다.

플러그인 이 설치되었으면 Installed 항목에서 O표시가, 제거되었으면 X표시가 되며, 현재 설치되어 있는 플러그인 다음과 같은 방법으로 확인할 수 있다.

Web타입 프로젝트의 경우 해당 플러그인과 샘플을 함께 설치할 수 있으며, 해당 플러그인 적용된 샘플은 프로젝트를 실행시켜봄으로써 확인할 수 있다. Service 타입 프로젝트의 경우 해당 플러그인의 라이브러리만 설치된다.

표 15.1. 예) 플러그인 설치 결과

	web 타입 프로젝트	service 타입 프로젝트
ant 빌드	지정한 패키지 경로에 설치한 플러그인 샘플 소스 코드 생성, src/main/webapp/WEB-INF/lib에 플러그인 관련 라이브러리 설치 또는, 라이브러리만 설치하고자 한 경우, src/main/webapp/WEB-INF/lib에 플러그인 관련 라이브러리 설치	프로젝트가 생성된 위치 하위의 lib 폴더에 플러그인 관련 라이브러리 설치
maven 빌드	pom.xml 에 플러그인 관련 dependency 추가	pom.xml 에 플러그인 관련 dependency 추가

샘플 실행방법은[New Project Creation]의 프로젝트 실행방법 내용을 참조한다.

아래는 플러그인 설치 및 제거와 관련하여 정상적으로 반영되었는지 확인하는 방법이다.

- metadata 정보확인: 프로젝트폴더 >> .metadata폴더>> plugins.xml 파일에 list로 관리한다.

```
<plugin>
  <checked>false</checked>
  <srcInstalled>true</srcInstalled>
  <customed>false</customed>
  <samples>true</samples>
</plugin>
<plugin>
  <checked>false</checked>
  <name>miplatform</name>
  <groupId>anyframe.plugin</groupId>
  <artifactId>anyframe.plugin.miplatform</artifactId>
  <version>4.5.0</version>
  <installed>true</installed>
  <srcInstalled>true</srcInstalled>
  <customed>false</customed>
  <samples>true</samples>
</plugin>
```

- Ant 프로젝트

- Web 타입 프로젝트: 플러그인 추가/삭제 시 관련된 라이브러리는 프로젝트 Web App Libraries 에 추가/삭제
- Service 타입 프로젝트: Service 타입 프로젝트의 .classpath에 설치된 플러그인 정보가 표시된다. 아래는 Service 타입 프로젝트의 .classpath일부분이다.

```
중략...
<!--hibernate here -->
<!--hibernate-START-->
  <classpathentry exported="true" kind="var"
    path="GEN_REPO/antlr/antlr/2.7.7/antlr-2.7.7.jar" />
  <classpathentry exported="true" kind="var"
    path="GEN_REPO/anyframe/anyframe.hibernate/4.5.0/
    anyframe.hibernate-4.5.0.jar" />
  <classpathentry exported="true" kind="var"
    path="GEN_REPO/dom4j/dom4j/1.6.1/dom4j-1.6.1.jar" />
  <classpathentry exported="true" kind="var"
    path="GEN_REPO/net/sf/ehcache/ehcache/1.6.2/ehcache-1.6.2.jar" />
중략...
<!--hibernate-END-->
```

- Maven 프로젝트: 설치한 플러그인과 관련된 라이브러리 정보가 pom.xml 의 <dependency>에 추가됨

- 4 플러그인 Install/Uninstall 시에 다음 그림과 같은 메시지가 뜨는 경우는 tomcat이 실행된 상태에서 tomcat이 인식하고 있는 파일에 변경이 있는 경우에 발생하는 것이다.

아래와 같은 메시지는 플러그인 Install/Uninstall 전에 tomcat 동작을 stop 시킬 경우에는 뜨지 않는다.



참고

위에서 설명한 기능 수행 도중에는 프로젝트 생성, 코드 생성 및 DB 변경 등과 같은 Anyframe Gen의 기능을 동시에 수행시킬 수 없으므로 유의하도록 한다. (Background 실행을 통한 기능 수행 시)

16.CTIP

Gen에서 Hudson을 연결하여 Hudson에 job을 등록하고 job을 실행하기 위해서 사용한다. Hudson 서버와 통신을 하기 위해서는 anyframe.gen.eclipse.hudson-X.X.X.jar 파일을 이용해 설정을 해야 한다. 설정과 관련된 자세한 사항은 [Installation]을 참조한다.

1 프로젝트를 선택한 후 우 클릭 >> **Anyframe Gen** 을 선택한다.

2 Anyframe Gen Editor에서 CTIP Tab을 선택하면 다음과 같은 항목이 보인다.

- Hudson URL



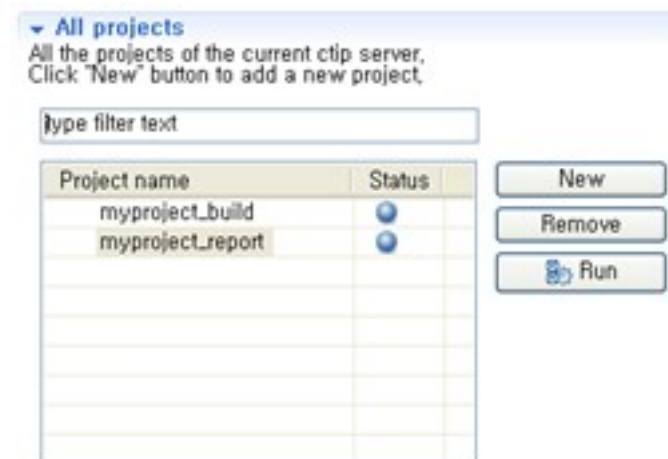
- URL: Hudson 서버에 접근하기 위한 url
- Configure...: Hudson 서버에 접근하기 위한 url을 등록,수정,삭제 등 관리하기 위한 팝업 화면 호출

- Hudson Configuration



- MAVEN_HOME: Maven이 설치된 경로로 Hudson 서버가 start된 상태일 때만 활성화됨
- Hudson URL in Email: Hudson Email Notification 시 사용할 url

- All projects



- Project name: ctip job name
- Status: job 수행 상태로, job을 빌드 하는 중일 경우 build...라는 메시지가 출력됨
- New : 새로운 job 추가
- Remove : job 삭제
- Run : job 실행

- Project Details

▼ Project Details
Define, review and modify the selected project configuration.
Required fields are denoted by *.

Build Type* build report

Project Name*

Custom Workspace

SCM Server Type

SCM Server URL

Build schedule

Build other project

- Build Type* : build는 빌드를 수행하고 배포파일 생성, report는 build 기능에 JUnit test, Emma, Jdepend, PMD등의 reporting 기능도 수행

표 16.1. Build Type

	build	report
Ant(target)	deploy	all
Maven(goal)	package	clean site package

- Project Name* : Hudson job name
- Custom Workspace : workspace 경로지정
- SCM Server Type : subversion, cvs, none 세가지 type 중 선택
- SCM Server URL : SCM 서버 url
- Build schedule : job 실행 스케줄 정의
- Build other project : 해당 job 수행 후 실행한 job의 이름 지정

3 빌드를 수행할 job을 목록에서 선택하고 Run 버튼을 클릭하여 빌드를 수행한다.

빌드가 수행되면 Status는 "build..."로 보여진다. 빌드 Status에 대해 scheduling이 걸려있지 않기 때문에, 빌드 결과는 **refresh** 버튼을 클릭하여 확인한다.

빌드가 다 수행이 되면 refresh 버튼을 눌렀을때 빌드 결과에 대한 이미지가 보여진다. 성공은 파란색, 실패는 빨간색.

Web타입 프로젝트의 경우 해당 플러그인과 샘플이 함께 설치되며, 해당 플러그인 적용된 샘플은 프로젝트를 실행시켜봄으로써 확인할 수 있다. Service 타입 프로젝트의 경우 해당 플러그인의 라이브러리만 설치된다.



참고

위 화면을 통해서 test case 가 없는 프로젝트에 대해서 job을 생성한 후 빌드했을 때, FAILURE가 발생한다. 이는 에러는 아니지만 test case에 대해 recoding을 하기 위해, junit, emma 플러그인이 출력하는 메시지이므로 test case를 사용하지 않을 경우 해당 플러

그인을 사용하지 않도록 설정해주어야 한다. "[프로젝트명]_report" job의 configure에서 "Publish JUnit test result report", "Record Emma coverage report" action을 uncheck 한다.

17. Project Build

Anyframe Gen 을 이용해 프로젝트 빌드를 수행할 수 있다.

프로젝트 생성 시에 빌드 스크립트 파일이 함께 생성되어, 빌드를 수행할 수 있다. 빌드 도구로 Ant를 사용할 경우 디폴트 타겟 빌드 수행 시 컴파일, 테스트, 패키징, 배포 등의 공통적인 빌드 프로세스 단계를 수행해주고 있다. Maven 프로젝트의 경우 POM 파일을 이용하여 빌드 및 패키징, 리포팅을 수행할 수 있다.

아래의 두 가지 경우로 나누어서 살펴보자.

- [Ant 프로젝트 빌드]
- [Maven 프로젝트 빌드]

17.1. Ant 프로젝트 빌드

1. 프로젝트 내 build.xml 확인

프로젝트 내부로 자동 생성되어진 build.xml 파일이 존재한다. 빌드 스크립트 내에는 빌드 수행을 위한 compile, test, package 등 공통적으로 사용되는 빌드 타겟들이 존재한다. 실제 구동 내역을 가지고 있는 공통 스크립트는 [Gen Home]\cli\scripts\project-build.xml 파일 내에 작성되어 있다. 프로젝트 별로 변경해야 하는 일이 생기면 공통 스크립트를 수정하면 된다. 또한 각 프로젝트 단위 별로 공통 빌드 내역 외에 특화된 빌드 내용을 추가해야 하는 경우에는 해당 빌드 타겟을 확장하여 추가해 넣을 수 있다.

web 타입 프로젝트의 경우, [프로젝트가 생성된 위치]src/main/webapp/WEB-INF/lib 폴더를 클래스패스로 설정하여 빌드하고, service 타입 프로젝트의 경우, [프로젝트가 생성된 위치]/lib 폴더를 클래스패스로 설정하여 빌드한다.

다음은 Web 타입 프로젝트에 생성된 build.xml 의 일부분이다.

```
<project name="Build module" default="default" basedir=". ">
  <property file = "build.properties"/>
  <import file = "${gen.home}/cli/scripts/project-build.xml"/>

  <target name="clean" depends = "shared:clean" />
  <target name="init" depends = "shared:init" />
  <target name="resources" depends = "shared:resources" />
  <target name="compile" depends = "shared:compile" />
  <target name="test-resources" depends = "shared:test-resources" />
  <target name="test-compile" depends = "shared:test-compile" />
  <target name="package" depends = "shared:package" />
  <target name="extract" depends = "shared:extract" />
  <target name="emma-jars" depends = "shared:emma-jars" />
  <target name="test" depends = "shared:test" />
  <target name="report" depends = "shared:report" />

  <target name = "default" depends = "init, resources, compile, package"/>
  <target name = "deploy" depends = "default, extract"/>
  <target name = "all" depends = "clean, init, resources, compile,
    test-resources, test-compile, package, emma-jars, test, report"/>
</project>
```

2. 프로젝트 내 build.xml 선택 후 >> Run As >> Ant Build 을 선택

디폴트 타겟이 수행되어 다음과 같은 빌드 프로세스를 거치게 된다. 최종 결과물인 프로젝트 바이너리 파일은 repo 폴더 하위에 존재하게 된다.

init -> resources -> compile -> package -> deploy

3. 프로젝트 내 **build.xml** 선택 후 >> **Run As >> Ant Build...** 을 선택

디폴트 타겟이 아닌 리포팅 결과물을 생성해내는 타겟을 실행시켜 본다. 빌드 시 코드 분석 리포트, 테스트 수행 결과 리포트, 코드 커버리지 리포트 등을 생성시킨다. 리포트 결과물은 프로젝트 내 output 폴더에 생성되므로 빌드가 모두 완료된 뒤 확인해보도록 한다.

clean -> init -> resources -> compile -> test-resources -> test-compile -> package -> emma-jars -> test -> report

17.2.Maven 프로젝트 빌드

프로젝트 내 pom.xml 확인

Maven은 pom.xml 파일을 기반으로 빌드되기 때문에 파일에 정의되어 있는 내용 확인한다. Reporting을 위해서는 <reporting>에 report를 남기고 싶은 플러그인들이 정의되어 있어야 하며, Anyframe Gen을 통해서 생성된 Maven 프로젝트에는 JUnit Test, Emma, Jdepend, PMD 등의 reporting 플러그인이 추가되어 있다.

다음은 Web 타입 프로젝트에 생성된 build.xml 의 일부분이다.

```
<reporting>
  <excludeDefaults>true</excludeDefaults>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-surefire-plugin</artifactId>
      <version>2.5</version>
      <configuration>
        <forkMode>always</forkMode>
      </configuration>
    </plugin>
    중략...
  </reporting>
```

프로젝트 내 pom.xml 선택 후 >> Run As >> Maven Build...을 선택

Goals 에 **clean package** 를 수행하게 되면 빌드 수행결과 및 War 파일이 생성되고, **clean site package** 를 수행하게 되면 리포트 결과물도 함께 생성되게 된다. 빌드 결과물은 [프로젝트 Home]/target에 존재하며, 리포트 결과물은 [프로젝트 Home]/target/site/내에 존재한다.

만약 M2Eclipse 를 이용하여 이를 수행하였을 때 "site" goal이 정상적으로 수행이 안 된다면, pom.xml 파일이 있는 경로에서 위와 같은 goal을 cmd창을 통해 수행한다.

V. Templates Extensions

Anyframe Gen 틀을 통해 자동 생성되는 프로젝트 구조와 소스 코드는 각 개발자 혹은 프로젝트 상황에 맞게 확장하여 사용할 수 있는 구조를 제공한다. 템플릿 파일들의 배포 위치는 [Gen Home] 폴더 하위의 templates 폴더이며 Anyframe Gen 설치 시 기본적으로 default, map, miplatform 템플릿을 제공하고 있다.

기본적으로 제공되는 각각의 템플릿에 대해서 살펴본 후, 필요한 템플릿에 대해서 확장해보도록 한다.

표 28. 기본 템플릿

template name	description
default	가장 기본이 되는 템플릿으로 spring/spring mvc 기반으로 Business Layer와 Presentation Layer 간의 데이터 통신을 DTO(Data Transfer Object) 객체를 이용하여 구현하며, 화면은 JSP 페이지로 구성되어 있다.
map	spring 기반으로 Business Layer와 Presentation Layer 간의 데이터 통신을 Map(java.util.Map) 객체를 이용하도록 Business Layer 코드, Presentation Layer 코드를 생성해준다. 또한 DAO 코드는 Query Service를 이용하여 생성해주고 있다. Hibernate를 이용한 DAO 코드는 생성하지 않는다.
miplatform	spring/spring mvc 기반으로 Business Layer와 Presentation Layer 간의 데이터 통신을 MiPlatform의 객체(DatasetList, VariableList)를 이용하여 구현하며, 화면은 MiPlatform을 이용한 UI로 구성되어 있다.

18. 프로젝트 템플릿 확장

[Gen Home] 폴더 하위의 templates 폴더를 살펴보면 default, map, miplatform 폴더가 존재하고 그 하위로 각각 project와 source 폴더가 존재하고 있음을 확인할 수 있다.

프로젝트 템플릿을 확장한다는 것은 Anyframe Gen을 통해 생성되는 프로젝트 구조를 포함하여 샘플 코드, 프로젝트 내부 빌드 파일 등을 각 개발자 혹은 프로젝트 상황에 맞게 수정하여 사용할 수 있다는 것으로 여기서는 Maven의 Archetype을 작성함으로써 프로젝트 템플릿을 확장할 수 있다. Archetype에 대한 자세한 내용은 Introduction to Archetypes [<http://maven.apache.org/guides/introduction/introduction-to-archetypes.html>] 을 참조하도록 한다.

어플리케이션의 프로젝트 구조가 Web 타입 인지 Service 타입 인지에 따라 Project Template 확장 방법이 다르므로 유의하도록 한다.

18.1. Web 타입 프로젝트 템플릿 확장

Web 타입 프로젝트의 경우, 기본적인 프로젝트 구조 및 샘플 코드가 Anyframe의 Foundation Plugin(anyframe.plugin.foundation-4.5.0.jar)으로부터 생성된다.

Anyframe Gen을 통해서는 샘플 코드 일부와 빌드 스크립트가 생성되므로 이 매뉴얼에서는 Anyframe Gen을 통해 변경될 수 있는 부분을 설명하겠다. Anyframe Foundation Plugin에 대한 설명은 Anyframe 매뉴얼을 참고하도록 한다.

아래 나열한 순서에 따라서 확장해보도록 한다. 아래에서는 message source 파일(properties)을 더 추가해보는 예제로 구성되어 있다.

1 Anyframe source code repository(Subversion)으로부터 anyframe.gen.archetype.singleconfig 프로젝트(SVN URL:<http://dev.anyframejava.org/subv/anyframe-opensource/gen/tags/gen-1.5.1/dev/archetype/anyframe.gen.archetype.singleconfig>)를 내려받는다. Subversion에 Access하는 자세한 방법은 이곳(Subversion 연결) [<http://www.anyframejava.org/development/subversion>] 을 참고하도록 한다.

2 Eclipse에 anyframe.gen.archetype.singleconfig 프로젝트를 Import한 후, 프로젝트 명을 변경시킨다. 이때 이름은 반드시 **anyframe.gen.archetype.singleconfig**로 시작 하는 이름이어야 한다.

예를 들어, anyframe.gen.archetype.singleconfig.ext로 프로젝트 명을 변경시켜보자.

- Eclipse Project 명 변경: 프로젝트 선택 후 마우스 우측 버튼 선택 > Refactor > Rename...> anyframe.gen.archetype.singleconfig.ext 입력 > OK
- Maven Project 명 변경: 프로젝트 하위에 있는 pom.xml 파일을 Open하여 아래와 같이 groupId, artifactId, version 정보를 원하는 정보로 변경하도록 한다.

```
<groupId>templates.project.ext</groupId>
<artifactId>anyframe.gen.archetype.singleconfig.ext</artifactId>
<version>1.0.0</version>
```

- Project 폴더명까지 모두 anyframe.gen.archetype.singleconfig.ext으로 변경하여 기존의 anyframe.gen.archetype.singleconfig 프로젝트와 혼동되지 않도록 관리한다.

3 anyframe.gen.archetype.singleconfig.ext 프로젝트 하위의 src/main/resources/archetype-resources 폴더 내 파일들을 변경 및 추가할 수 있다.

샘플 코드 혹은 빌드 스크립트 파일을 변경할 수 있다. 샘플 코드는 src/main/resources/archetype-resources/src 폴더 하위에, 빌드 스크립트 파일(build.xml)은 src/main/resources/archetype-resources/pjt 폴더 하위에 위치하고 있다.

- 4 신규 message resource 파일을 추가하여, 프로젝트 생성 시 나타나도록 message-ext.properties 파일을 추가한다.

그러므로 src/main/resources/archetype-resources/src/main/resources/message 폴더 하위에 message-ext.properties 파일을 추가한다.

- 5 수정 및 추가한 정보에 의해 부가적으로 변경되어야 하는 파일이 존재하는 경우, 함께 수정하도록 한다.

message resource 파일이 추가되면, spring messageSource bean에 properties 파일을 추가 설정해 주도록 한다. 즉, src/main/resources/archetype-resources/src/main/resources/spring/context-message.xml 파일에 다음과 같이 value 태그를 추가해 넣는다.

```
<bean id="messageSource"
  class="org.springframework.context.support.ResourceBundleMessageSource">
  <property name="basenames">
    <list>
      <value>message/message-ext</value>
      <value>message/message-generation</value>
      <value>message/message-productmgmt</value>
      <value>anyframe/core/properties/messages/properties</value>
      중략...
    </list>
  </property>
</bean>
```

- 6 확장된 anyframe.gen.archetype.singleconfig.ext 프로젝트를 패키징한다.

Maven으로 프로젝트를 빌드하여 jar 파일로 패키징하도록 한다. Maven 빌드를 하기 위해서는 Maven을 먼저 설치해야 하는데 이는 Anyframe 매뉴얼 중 Maven 설치 및 환경 설정 부분 [http://dev.anyframejava.org/docs/anyframe/4.5.0/reference/htmlsingle/anyframe.html#installation_maven_sec001]을 참고하도록 한다. Maven 설치 및 환경 설정 한 이 후 "mvn package" command를 수행시킨다. 그 결과 패키징된 파일은 anyframe.gen.archetype.singleconfig.ext 프로젝트 하위의 target 폴더에 anyframe.gen.archetype.singleconfig.ext-1.0.0.jar 파일로 생성된다.

```
mvn package
```

- 7 확장된 anyframe.gen.archetype.singleconfig.ext 배포 파일을 [Gen Home] 폴더 하위의 templates 폴더에 위치시키는데 이때, 새로운 템플릿으로 명명하여 사용할 것이므로 templates 폴더 하위에 ext 와 같이 템플릿명을 나타내는 폴더를 생성한 뒤 다음과 같이 ext 폴더 하위로 project 폴더를 두고, anyframe.gen.archetype.singleconfig.ext-1.0.0.jar 파일을 배포한다.

즉, [Gen Home]/templates/default 폴더 전체를 복사하여 [Gen Home]/templates/ext 폴더에 위치시킨 후 기존의 anyframe.gen.archetype.singleconfig-1.0.0.jar 파일을 project 폴더에서 삭제하고 새로 패키징한 anyframe.gen.archetype.singleconfig.ext-1.0.0.jar 파일을 위치시킨다.

- 8 Web 타입 프로젝트 생성 기능을 수행시켜서 위에서 추가한 message-ext.properties 파일이 자동 생성되어 구성되는지 확인해보도록 한다.

CLI상에서 Web 타입 프로젝트 생성 시 build.properties 파일의 template_type 항목의 값을 ext로 변경하고 생성하도록 한다. Eclipse Plugin을 이용하여 Web 타입 프로젝트 생성 시에는 AnyframeGen Editor의 Configuration Tab 내에서 template을 ext로 변경하고 생성하도록 한다.

18.2.Service 타입 프로젝트 템플릿 확장

Service 타입 프로젝트의 경우, 자바 프로젝트 구조가 구성되어 생성된다.

다음 표와 같이 여러 종류의 Archetype이 있으므로 확장하고 싶은 Archetype에 대해 작업하도록 한다.

표 18.1. Service 타입 Projects Archetypes

archetype	description
anyframe.gen.archetype.service.annotation	프로젝트의 구조를 생성시켜주는 archetype으로 Aspect 클래스 및 spring 설정 파일 그리고 Ant 빌드 프로젝트 시에 빌드 스크립트(build.xml)를 제공한다.

아래 나열한 순서에 따라서 확장해보도록 한다.

- 1 Anyframe source code repository(Subversion)으로부터 archetype 프로젝트를 내려받도록 한다. (SVN URL은 다음 표를 참조한다.) Subversion에 Access하는 자세한 방법은 이곳(Subversion 연결) [<http://www.anyframejava.org/development/subversion>] 을 참고하도록 한다.

표 18.2. Archetype SVN URL

archetype	SVN URL
anyframe.gen.archetype.service.annotation	http://dev.anyframejava.org/subv/anyframe-opensource/gen/tags/gen-1.5.1/dev/archetype/anyframe.gen.archetype.service.annotation

- 2 Eclipse에 archetype 프로젝트를 Import한 후, 프로젝트 명을 변경시킨다. 이때 이름은 반드시 **anyframe.gen.archetype.service**로 시작 하는 이름이어야 한다.

예를 들어, anyframe.gen.archetype.service.annotation.ext로 프로젝트 명을 변경시켜보자.

- Eclipse Project 명 변경: 프로젝트 선택 후 마우스 우측 버튼 선택 > Refactor > Rename...> anyframe.gen.archetype.service.annotation.ext 입력 > OK
- Maven Project 명 변경: 프로젝트 하위에 있는 pom.xml 파일을 Open하여 아래와 같이 groupId, artifactId, version 정보를 원하는 정보로 변경하도록 한다.

```
<groupId>templates.serviceproject.ext</groupId>
<artifactId>anyframe.gen.archetype.service.annotation.ext</artifactId>
<version>1.0.0</version>
```

- Project 폴더명까지 모두 anyframe.gen.archetype.service.annotation.ext으로 변경하여 기존의 anyframe.gen.archetype.service.annotation 프로젝트와 혼동되지 않도록 관리한다.
- 3 anyframe.gen.archetype.service.annotation.ext 프로젝트 하위의 src/main/resources/archetype-resources 폴더 내 파일들을 변경 및 추가할 수 있다.

샘플 코드 혹은 빌드 스크립트 파일을 변경할 수 있다. 샘플 코드는 src/main/resources/archetype-resources/src 폴더 하위에, 빌드 스크립트 파일(build.xml)은 src/main/resources/archetype-resources/pjt 폴더 하위에 위치하고 있다.

- 4 LoggingAspect 클래스에 afterLogging 메소드를 추가한다.

아래와 같이 src/main/java/common/aspect/LoggingAspect.java 파일을 열어서 afterLogging 메소드를 추가해넣는다. 메소드 호출 후에도 로깅을 남기도록 수정한다.

```
@Inject
private MessageSource messageSource;
중략...
@After("serviceMethod()")
public void afterLogging(JoinPoint thisJoinPoint) {
    Class clazz = thisJoinPoint.getTarget().getClass();
```

```
String className = clazz.getName();
String classSimpleLowerName = thisJoinPoint.getTarget().getClass().
    getSimpleName().toLowerCase();
String methodName = thisJoinPoint.getSignature().getName();
String methodLowerName = methodName.toLowerCase();

StringBuffer buf = new StringBuffer();
buf.append("\n** After Logging Aspect : executed " + methodName + "() in "
    + className + " Class.");
buf.append("\n*****\n");
buf.append(messageSource.getMessage("success." + classSimpleLowerName + "."
    + methodLowerName, new String[] {}, "no messages", Locale.getDefault()));
buf.append("\n*****\n");

Log logger = LogFactory.getLog(clazz);
if (logger.isDebugEnabled())
    logger.debug(buf.toString());
}
```

5 확장된 anyframe.gen.archetype.service.annotation.ext 프로젝트를 패키징한다.

Maven으로 프로젝트를 빌드하여 jar 파일로 패키징하도록 한다. Maven 빌드를 하기 위해서는 Maven을 먼저 설치해야 하는데 이는 Anyframe 매뉴얼 중 Maven 설치 및 환경 설정 부분 [http://dev.anyframejava.org/docs/anyframe/4.5.0/reference/htmlsingle/anyframe.html#installation_maven_sec001] 을 참고하도록 한다. Maven 설치 및 환경 설정한 이후 "mvn package" command를 수행시킨다. 그 결과 패키지 파일은 anyframe.gen.archetype.service.annotation.ext 프로젝트 하위의 target 폴더에 anyframe.gen.archetype.service.annotation.ext-1.0.0.jar 파일로 생성된다.

```
mvn package
```

6 확장된 anyframe.gen.archetype.service.annotation.ext 배포 파일을 [Gen Home] 폴더 하위의 templates 폴더에 위치시키는데 이때, 새로운 템플릿으로 명명하여 사용할 것이므로 templates 폴더 하위에 ext와 같이 템플릿명을 나타내는 폴더를 생성한 뒤 다음과 같이 ext 폴더 하위로 project 폴더를 두고, anyframe.gen.archetype.service.annotation.ext-1.0.0.jar 파일을 배포한다.

즉, [Gen Home]/templates/default 폴더 전체를 복사하여 [Gen Home]/templates/ext 폴더에 위치시킨 후 기존의 anyframe.gen.archetype.service.annotation-1.5.1.jar 파일을 project 폴더에서 삭제하고 새로 패키징한 anyframe.gen.archetype.service.annotation.ext-1.0.0.jar 파일을 위치시킨다.

7 Service 타입 프로젝트 생성 기능을 수행시켜서 위에서 추가한 LoggingAspect.java 파일이 자동생성되어 구성되는지 확인해보도록 한다.

CLI상에서 Service 타입 프로젝트 생성 시 build.properties 파일의 template_type 항목의 값을 ext로 변경하고 생성하도록 한다. Eclipse Plugin을 이용하여 Service 타입 타입 프로젝트 생성 시에는 AnyframeGen Editor의 Configuration Tab 내에서 template을 ext로 변경하고 생성하도록 한다.

19.소스 코드 템플릿 확장

[Gen Home] 폴더 하위의 templates 폴더를 살펴보면 아래와 같이 default,map, miplatform 폴더가 존재하고 그 하위로 각각 project와 source 폴더가 존재하고 있음을 확인할 수 있다.

소스 코드 템플릿을 확장한다는 것은 Anyframe Gen을 통해 생성되는 도메인 클래스, CRUD 소스 코드에 대해서 각 개발자 혹은 프로젝트 상황에 맞게 수정하여 사용할 수 있다는 것으로 여기서는 FreeMarker Template을 수정함으로써 소스 코드 템플릿을 확장할 수 있다. FreeMarker에 대한 자세한 내용은 FreeMarker 사이트 [<http://freemarker.org/>] 를 참조하도록 한다.

프로젝트가 Web타입 인지 Service타입 인지에 상관없이 소스 코드 템플릿 확장 방법은 동일하다.

19.1.템플릿 폴더 생성

FreeMarker Template(*.ftl) 파일들을 수정하여 자동 생성되는 소스 코드에 대한 템플릿을 확장해보도록 한다.

default 템플릿을 확장하여 ext 템플릿을 구성해보자. 이때 프로젝트 템플릿은 확장하지 않고 소스 코드 템플릿만 확장해본다. 프로젝트 템플릿 확장하는 방법은 [프로젝트 템플릿 확장 매뉴얼] 을 참고하도록 한다.

[Gen Home]/templates/default 폴더 전체를 복사한 후 이름을 ext로 변경시켜서 [Gen Home]/templates/ext 폴더를 만들고 [Gen Home]/templates/ext/source 폴더 하위의 template.config 파일이 있는지 확인한다.

19.2.템플릿 파일 목록 작성 (template.config)

template.config 파일을 열고 소스 코드 생성에 필요한 템플릿 파일 목록을 작성한다.

template.config 파일 내 작성되는 정보는 소스 코드 생성 시 필요한 코드 템플릿 목록 정보로 default 템플릿을 통해 제공되는 템플릿 정보는 다음 표와 같다.

타입 별 템플릿 파일에 대한 설명으로, template.config 파일 내 템플릿 목록으로 작성되어야 CRUD 소스 코드 생성 시 결과 코드로 생성된다. service 타입 템플릿은 Business Layer 코드 생성 시 사용되는 템플릿이고, web 타입 템플릿은 Presentation Layer 코드 생성 시 사용된다.

template.config 파일 내 템플릿 정보 등록 시 아래와 같은 항목 내용을 작성하도록 한다.

- type: service, web 타입
- ftl: freemarker template 파일
- src: 생성될 소스 코드 파일 명
 - {basepkg-name} : 대표패키지
 - {class-name} : 엔티티 클래스 명
- mergeSrc: 생성된 소스 코드를 반영할 파일명
- mergeKey: 생성된 소스 코드를 반영할 위치정보 key
- share: sample db data 공유 여부를 true, false로 설정
- dao: hibernate, query 중 dao framework 설정
- description: 템플릿 설명

표 19.1. Service Type template configuration

type	ftl	src	mergeSrc	mergeKey	share	dao	description
service	dao/sample-data.ftl	merge/{class-name}-sample-data.xml	/src/test/resources/sample-data.xml	!-{class-name}-START-	false	N/A	test sample db data
service	service/service-test.ftl	src/test/java/{basepkg-name}/service/{class-name}ServiceTest.java			true	N/A	service 에 대한 통합 test case
service	service/service.ftl	src/main/java/{basepkg-name}/service/{class-name}Service.java			false	N/A	service interface class
service	service/service-impl.ftl	src/main/java/{basepkg-name}/service/impl/{class-name}ServiceImpl.java			false	N/A	service implementation class
service	dao/hibernate/dao.ftl	src/main/java/{basepkg-name}/service/impl/{class-name}Dao.java			false	hibernate	hibernate dao implementation class
service	dao/hibernate/dynamic-hibernate/hibernate-dynamic-hibernate-{class-name}.xml	src/main/resources/hibernate-dynamic-hibernate-{class-name}.xml			false	hibernate	hibernate dynamic hql mapping xml file
service	dao/query/dao.ftl	src/main/java/{basepkg-name}/service/impl/{class-name}Dao.java			false	query	query dao implementation class
service	dao/query/mapping-query.ftl	src/main/resources/sql/mapping-query-{class-name-lower}.xml			false	query	query service mapping xml file
service	web/ApplicationResource	merge/{class-name}-ApplicationResource	/src/main/resources/message-generation.properties	# - {class-name}-START	false	N/A	message source properties 파일

표 19.2. Web Type template configuration

type	ftl	src	mergeSrc	mergeKey	share	dao	description
web	web/ spring/ controller/ test.ftl	src/test/java/ {basepkg- name}/web/ {class- name}Controller Test.java			false	N/A	spring mvc controller class test case
web	web/ spring/ controller/ main.ftl	src/main/java/ {basepkg- name}/web/ {class-name} Controller.java			false	N/A	spring mvc controller class
web	web/ spring/ list- view.ftl	src/main/ webapp/WEB- INF/jsp/ generation/ {class-name- lower}/list.jsp			false	N/A	list view jsp page
web	web/ spring/ form- view.ftl	src/main/ webapp/WEB- INF/jsp/ generation/ {class-name- lower}/form.jsp			false	N/A	detail view jsp page
web	web/ menu.ftl	merge/{class- name}- menu.jsp	/src/main/ webapp/ sample/ layouts/gen/ left-gen.jsp	!-{class-name}- START-	false	N/A	menu jsp page
web	web/ tiles- menu.ftl	merge/{class- name}-tiles- menu.jsp	/src/main/ webapp/WEB- INF/ tilesviews.xml	!-{class-name}- START-	false	N/A	tiles menu xml

19.3. 템플릿 파일(*.ftl) 수정

[Gen Home]/templates/ext/source 폴더 하위에서 확장하고 싶은 ftl 파일들을 선정하여 수정하도록 한다.

source 폴더 하위에 있는 ftl 파일들은 원하는 폴더로 묶어서 관리할 수 있다. 현재에는 다음 표와 같이 4개의 폴더(model, service, dao, web)로 구분되어 있다. 자유롭게 변경할 수 있다. 단, 폴더명과 ftl 파일명 변경 시 위에서 설명한 template.config 파일 내 ftl 항목 내용도 반드시 변경시켜줘야 한다.

표 19.3. template folder

folder name	description
model	도메인 클래스 생성 시 사용되는 ftl들 존재
service	service 클래스/테스트 케이스 및 Spring bean 설정 파일 ftl들 존재
dao	샘플 데이터, hibernate/query dao 클래스 및 Spring bean 설정 파일 ftl들 존재
web	메뉴, message resource, spring mvc 클래스/테스트 케이스 및 Spring bean 설정 파일, JSP ftl들 존재

예를 들어 service 생성 시 기본 CRUD 메소드 외 다른 메소드를 추가해보자.

- [Gen Home]/templates/ext/source/service/service.ftl 파일을 열어 다음과 같이 getProcedureList 메소드를 추가한다.

getProcedureList의 리턴 타입이 List 이므로 클래스 상단에 import 구문을 추가해줘야 함에 유의하도록 한다.

```
package ${basepackage}.service;

import java.util.List;

import anyframe.common.Page;
import anyframe.common.util.SearchVO;
중략...
import ${pojo.packageName}.${pojo.shortName};

public interface ${pojo.shortName}Service{

    void create(${pojo.shortName} ${pojoNameLower}) throws Exception;
    List getProcedureList(SearchVO searchVO) throws Exception;
}
```

이 ftl 구문 내에서 클래스, 메소드 등의 이름을 변경하거나 내용을 수정할 수 있다. service interface(service.ftl) 클래스에 대한 내용 수정 시 service implementation(service-impl.ftl) 클래스에 대한 내용도 함께 수정해줘야 함에 유의하도록 한다.

- [Gen Home]/templates/ext/source/service/service-impl.ftl 파일을 열어 다음과 같이 getProcedureList 메소드를 추가한다.

getProcedureList의 리턴 타입이 List 이므로 클래스 상단에 import 구문을 추가해줘야 함에 유의하도록 한다.

```
<#assign pojoNameLower = pojo.shortName.substring(0,1).toLowerCase()+
    pojo.shortName.substring(1)>
package ${basepackage}.service.impl;

import java.util.List;
import anyframe.common.Page;
import anyframe.common.util.SearchVO;
import ${basepackage}.service.${pojo.shortName}Service;
중략...
public class ${pojo.shortName}ServiceImpl implements ${pojo.shortName}Service {

    @Inject
    @Named("${pojoNameLower}Dao")
    ${pojo.shortName}Dao ${pojoNameLower}Dao;
```

```

public void create(${pojo.shortName} ${pojoNameLower}) throws Exception {
    this.${pojoNameLower}Dao.create(${pojoNameLower});
}

public List getProcedureList(SearchVO searchVO) throws Exception{
    중략...
}
}

```

19.4. 템플릿 파일(*.ftl) 변수

템플릿 파일(*.ftl)을 수정하기 위해서는 일반 문자열 자체를 변경하여 클래스나 메소드 명을 변경시킬 수도 있으나 템플릿 파일 내 사용되는 변수(\${변수명})를 이용하여 수정 시 좀더 다양하게 템플릿을 확장시킬 수 있다.

다음은 템플릿 파일 내에서 자주 사용되는 변수에 대한 설명을 표로 나타낸 것이다.

표 19.4. FTL 파일 내 변수 목록

name	description
pojo.shortName	CRUD 대상이 되는 도메인 클래스명(ex. Board)
pojoNameLower	CRUD 대상이 되는 도메인 클래스명의 소문자 형태(ex. board)
pojo.packageName	CRUD 대상이 되는 도메인 클래스의 패키지명
basepackage	소스 코드 생성 시 생성되는 클래스의 상위 패키지로 어플리케이션 대표 패키지명
pojo.identifierProperty.name	도메인 클래스의 Primary Key에 해당하는 속성명

다음은 아래와 같이 사용되는, 템플릿 파일 내에서 FreeMarker 구문을 통해 사용되는 로직에 대한 설명을 표로 나타낸 것이다. <#구문 내에서 사용된다.

foreach, if, lt, assign 등과 같은 FreeMarker 문법에 대한 내용은 FreeMarker 사이트 [<http://freemarker.org/>] 를 참조하도록 한다.

```

<#foreach field in pojo.getAllPropertiesIterator()>
  <#foreach column in field.getColumnIterator()>
    <#if field.equals(pojo.identifierProperty) && !column.nullable
      && !c2h.isCollection(field) && !c2h.isManyToOne(field)
      && !c2j.isComponent(field)>
      <#lt/><form:option value="${field.name}">
      <anyframe:message code="${pojoNameLower}.${field.name}"/>
      </form:option>
    </#if>
  </#foreach>
  <#if field.equals(pojo.identifierProperty) && !c2h.isCollection(field)
    && !c2h.isManyToOne(field) && c2j.isComponent(field)>
    <#lt/>
    <#assign pojoIdentifier = pojo.identifierProperty.getValue() >
    <#foreach idfield in pojoIdentifier.getPropertyIterator()>
      <#lt/><form:option value="${field.name}.${idfield.name}">
      <anyframe:message code="${pojoNameLower}.${field.name}.${idfield.name}"/>
      </form:option>
    </#foreach>
  </#if>
</#foreach>

```

표 19.5. FTL 파일 내 로직

logic	description
pojo.getAllPropertiesIterator()	도메인 클래스의 모든 속성 정보를 얻어냄
c2j.isComponent(pojo.identifierProperty)	도메인 클래스의 Primary Key에 해당하는 속성이 Composite Key 클래스로 구성되었는지 여부를 리턴함
c2j.isComponent(field)	도메인 클래스의 해당 field 속성이 primitive type이 아닌 클래스로 구성되었는지 여부를 리턴함
c2h.isCollection(field)	도메인 클래스의 해당 field 속성이 Collection 클래스로 구성되었는지 여부를 리턴함
c2h.isManyToOne(field)	도메인 클래스의 해당 field 속성이 many to one 관계에 있는지 여부를 리턴함
dbdata.getSampleDataSet(pojo)	DB 테이블 별 컬럼값에 해당하는 샘플 데이터 값을 얻어냄



참고

pojo(POJOClass), c2j(Cfg2JavaTool), c2h(Cfg2HbmTool), data(AnyframeDataHelper), dbdata(AnyframeDBData), util(StringUtils) 등은 템플릿 파일 내에서 각 클래스의 메소드를 호출하여 로직을 수행할 수 있도록 해주는 변수들로 Anyframe Gen 툴 내에서 설정해주고 있다. 현재 제공되는 템플릿 파일내에서 사용되고 있는 모습을 참조하여 작성하도록 한다.

19.5. 변경된 템플릿 파일(*.ftl) 적용

확장된 ftl 파일을 실제 CRUD 소스 코드 생성 시 사용해본다.

CLI 혹은 Eclipse Plugin 툴을 이용하여 CRUD 소스 코드를 생성시켜서 변경된 템플릿 파일이 정상적으로 반영되었는지 확인한다. 소스 코드 생성 시 에러가 발생하지 않고, 실제 어플리케이션을 구동시켜서 정상적으로 동작해야 한다.