

# Anyframe Idgen Plugin



Version 1.0.3

저작권 © 2007-2011 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. Introduction .....	1
II. Id Generation .....	2
1. UuidGenService .....	3
1.1. Implementation .....	3
2. SequencedGenService .....	4
2.1. Implementation .....	4
3. TableIdGenService .....	5
3.1. Implementation .....	5
4. How to use a Generation Strategy .....	7
4.1. MixPrefixStrategy .....	7
4.2. TimestampStrategy (1.0.1 이후) .....	7
4.3. Id Generation Strategy를 implements하는 방법 .....	8

---

# I.Introduction

idgen plugin은 Unique한 ID를 생성하기 위한 기능의 라이브러리들로 구성되어 있다.

## Installation

Command 창에서 다음과 같이 명령어를 입력하여 idgen plugin을 설치한다.

```
mvn anyframe:install -Dname=idgen
```

installed(mvn anyframe:installed) 혹은 jetty:run(mvn clean jetty:run) command를 이용하여 설치 결과를 확인해볼 수 있다.

## Dependent Plugins

Plugin Name	Version Range
core [ <a href="http://dev.anyframejava.org/docs/anyframe/plugin/essential/core/1.0.3/reference/htmlsingle/core.html">http://dev.anyframejava.org/docs/anyframe/plugin/essential/core/1.0.3/reference/htmlsingle/core.html</a> ]	2.0.0 > *

---

## II.Id Generation

시스템 개발 시 공통적으로 많이 쓰이는 기능 중의 하나로, 유일한 ID를 생성하기 위해 사용할 수 있다. 주로 데이터베이스에 새로운 레코드를 추가하기 위해 ID를 받거나 혹은 파일 시스템에 새로이 파일을 생성하고자 할때 유일한 파일명을 보장 받기 위해 유일한 ID가 필요하다. Id Generation 서비스에 대한 구현체는 3가지이며, 다음은 각 구현체별 사용법이다.

---

---

# 1.UUIdGenService

새로운 ID를 생성하기 위해 UUID 생성 알고리즘을 이용하여 16 바이트 길이의 ID를 생성한다. 다음과 같은 설정을 가질 수 있다.

Property Name	Description	Required	Default Value		
address	물리적 네트워크 공간 상의 유일성을 보장하기 위해 IP 주소나 네트워크 카드의 MAC 주소를 값으로 지정한다. IP 주소일 경우 필드간의 구분은 "."로 하고, MAC 주소일 경우는 ":"로 한다. 지정하지 않으면 UUIdGenService는 내부에서 MAC 주소를 랜덤하게 생성한다.	False	랜덤 생성		

## 1.1.Implementation

- **Configuration**

다음은 IdGenerationService에서 사용할 UUIdGenService와 그 속성을 정의한 context-idgen.xml 의 일부이다. MAC Address를 기반으로 유일한 Id를 생성하는 UUIdGenService Bean을 정의하고 있다.

```
<bean name="UUIdGenService" class="org.anyframe.idgen.impl.UUIdGenServiceImpl">
  <property name="address" value="00:00:F0:79:19:5B"/>
</bean>
```

- **TestCase**

다음은 앞서 정의한 속성 설정을 기반으로 하여 UUIdGenService로부터 유일한 id를 추출하는 IdgenServiceTest.java 코드의 일부이다.

```
@Inject
@Named("UUIdGenService")
IdGenService uuid;

/**
 * [Flow #-1] Positive : try to get next String id and BigDecimal id.
 *
 * @throws Exception
 *         fail to test
 */
@Test
public void testUUIdGenService() throws Exception {
    // 1. get next String id
    for (int i = 0; i < 10; i++) {
        assertNotNull(uuid.getNextStringId());
    }
    // 2. get next BigDecimal id
    for (int i = 0; i < 10; i++) {
        assertNotNull(uuid.getNextBigDecimalId());
    }
}
```

## 2.SequenceIdGenService

새로운 ID를 생성하기 위해 Database의 SEQUENCE 문을 사용하는 서비스이다. 다음과 같은 설정을 가질 수 있다.

Property Name	Description	Required	Default Value
dataSource	SequenceIdGenService를 사용하는 경우 필요하다. Database 연결을 위한 DataSourceService를 지정한다.	Y	N/A
useBigDecimals	long 타입의 ID 대신 BigDecimal 타입의 ID를 사용하고자 할 때 정의한다.	N	false
query	SEQUENCE SQL을 value로써 지정한다.(Database에 의존적이다.)	Y	N/A

### 2.1.Implementation

- **Configuration**

다음은 SequenceIdGenService의 속성을 정의한 context-idgen.xml 의 일부이다. SequenceIdGenService는 common\_datasource를 통하여 Connection 객체를 얻게 되며, IDGEN\_SEQ 라는 이름의 Sequence를 사용한다. 따라서, 다음 속성 기반의 SequenceIdGenService 실행을 위해서는 IDGEN\_SEQ라는 이름의 Sequence가 생성되어 있어야 한다.

```
<bean name="SequenceIdGenService" class="org.anyframe.idgen.impl.SequenceIdGenServiceImpl"
destroy-method="destroy">
  <property name="dataSource" ref="dataSource" />
  <property name="query" value="SELECT NEXT VALUE FOR IDGEN_SEQ FROM IDS"/>
</bean>
```

- **TestCase**

다음은 앞서 정의한 속성 설정을 기반으로 하여 SequenceIdGenService를 이용하여 유일한 id를 추출하는 IdgenServiceTest.java 코드의 일부이다.

```
@Inject
@Named("SequenceIdGenService")
IdGenService sequenceid;

/**
 * [Flow #-4] Positive Case : try to get next Long id
 *
 * @throws Exception
 *         fail to test
 */
@Test
public void testSequenceIdGenServiceTest() throws Exception {
    // 1. get id twice to compare with each other
    String id1 = sequenceid.getNextStringId();
    String id2 = sequenceid.getNextStringId();
    assertEquals("fail to get differenct id", true, !id1.equals(id2));
}
```

## 3. TableIdGenService

TableIdGenService는 ID를 관리하는 특정 테이블을 이용하여 유일한 ID를 얻는다. ID 관리를 위한 이 테이블은 table\_name(Char 또는 VARCHAR타입), next\_id(integer type)과 같이 두 칼럼을 필요로 한다. table\_name 칼럼에는 ID 발급 대상 테이블명을, next\_id 칼럼에는 ID 생성에 사용될 순번을 입력하도록 한다. 신규 id를 할당받은 후에 next\_id 칼럼의 값은 TableIdGenService를 통해 신규 id 값으로 변경된다. (단, 신규 id를 얻고자 할 시점에, ID 발급 대상 테이블과 관련된 순번이 ID 관리 테이블에 저장되어 있지 않으면 TableIdGenService는 ID 발급 대상 테이블 이름과 순번 '1'을 ID 관리 테이블에 추가하는 작업을 수행한다. ) 또한 Generation Strategy 을 적용할 경우, TableIdGenSimple Service는 신규 id에 정의된 Strategy를 적용한 결과값을 전달해준다. TableIdGenSimple Service는 다음과 같은 설정을 가질 수 있다.

Property Name	Description	Required	Default Value
dataSource	TableIdGenSimple Service를 사용하는 경우 필요하다. Database 연결을 위한 DataSourceService를 지정한다.	Y	N/A
strategy	Id Generation Strategy를 지정한다. 기본적으로 prefix, 신규 Id, fillChar를 조합한 Id를 전달하는 MixPrefix를 사용한다.	N	N/A
blockSize	ID를 발급할 때마다 매번 데이터베이스에 접속한다면 시스템 성능 저하를 가져오므로, 한번에 TableIdGenService 내에서 발급 받아올 ID의 개수를 정할 수 있다.	N	10
table	발급한 ID를 관리하기 위한 테이블 이름.	N	ids
key	어떤 것에 발급하는 ID인지를 저장하기 위한 키 이름. 예를 들어, Order와 Product, 두 가지에 ID를 발급할 상황이라면 하나는 key 속성값으로 "order", 또다른 하나는 "product"라고 지정할 수 있다. 이 경우 TableIdGenService을 구현 클래스로 하는 2개의 서로 다른 Bean이 정의되어야 하므로 속성 정의의 간편화를 위해서는 관련 메소드 호출시 발급 대상을 입력 인자로 전달할 수도 있다.	N	<input type="checkbox"/>
useBigDecimals	Long 타입의 ID 대신 BigDecimal 타입의 ID를 사용하고자 할 때 쓴다.	N	false

### 3.1. Implementation

- Configuration

다음은 TableIdGenService의 속성을 정의한 context-idgen.xml의 일부이다. TableIdGenService는 기본적으로 mixPrefix Strategy를 적용하고 있으며, Genre 테이블에 유일한 id를 제공하기 위해 IDS라는 ID 관리 테이블을 사용할 것이다. 다음 속성 기반의 TableIdGenService 실행을 위해서는 IDS라는 테이블에 [TABLE\_NAME:IDGEN\_MOVIE, NEXT\_ID:초기값]와 같은 정보가 추가되어 있다라고 전제해 보자.

```
<bean name="TableIdGenService" class="org.anyframe.idgen.impl.TableIdGenServiceImpl"
destroy-method="destroy">
  <property name="dataSource" ref="dataSource" />
  <property name="blockSize" value="1"/>
  <property name="table" value="IDS"/>
</bean>
```

- TestCase

다음은 앞서 정의한 속성 설정을 기반으로 하여 SequenceIdGenService를 이용하여 유일한 id를 추출하는 IdgenServiceTest.java 코드의 일부이다.

```
@Test
public void testSimpleTableId() throws Exception {
    // 1. get id twice to compare with each other
    String id1 = tableIdSimple.getNextStringId("IDGEN_MOVIE");
    String id2 = tableIdSimple.getNextStringId("IDGEN_MOVIE");
    assertEquals("fail to get different id", true, !id1.equals(id2));
}
```

---

## 4.How to use a Generation Strategy

TableIdGenService를 사용할 때, 유일한 id 생성 시 generation strategy를 적용할 수 있으며, 정의된 Prefix를 순번과 조합하는 MixPrefixStrategy와 정의된 포맷의 현재 시간과 순번을 조합하는 TimestampStrategy를 제공하고 있다. (1.0.1 이후) generation strategy는 아래와 같이 TableIdGenService를 통해 참조될 수 있도록 정의해 주어야 한다.

- **dependency injection**을 사용한 **strategy** 참조

```
<bean name="TableIdGenWithStrategyService"
  class="org.anyframe.idgen.impl.TableIdGenServiceImpl"
  destroy-method="destroy">
  <property name="dataSource" ref="dataSource" />
  <property name="strategy" ref="tableIdMixPrefix" />
  <property name="blockSize" value="1"/>
  <property name="table" value="IDS"/>
  <property name="key" value="IDGEN_MOVIE"/>
</bean>

<bean name="tableIdMixPrefix" class="org.anyframe.idgen.impl.strategy.MixPrefixStrategy">
  <property name="prefix" value="MV-" />
  <property name="cipers" value="5" />
  <property name="fillChar" value="*" />
</bean>
```

### 4.1.MixPrefixStrategy

Anyframe의 TableIdGenService에서는 org.anyframe.idgen.impl.strategy.MixPrefixStrategy라는 generation strategy를 제공한다. MixPrefixStrategy는 id generation strategy의 한 종류로, id 생성 시 TableIdGenService 실행을 통해 추출된 순번과 prefix를 조합한다. 예를 들어 prefix가 'TEST-', TableIdGenService를 통해 전달된 순번이 '12', fillChar가 '0', cipers가 5일 경우 전달된 id의 값은 prefix와 순번을 조합하여 'TEST-00012'가 된다. 이 때, 자릿수(cipers)가 5, 순번이 '12'이면 순번은 5자리로 구성하되 빈 부분은 fillChar로 정의된 문자열로 채우기 때문에 '00012'와 같은 형태가 된다.

```
<bean name="mixPrefix" class="org.anyframe.idgen.impl.strategy.MixPrefixStrategy">
  <property name="prefix" value="TEST-"/>
  <property name="cipers" value="5"/>
  <property name="fillChar" value="0"/>
</bean>
```

### 4.2.TimestampStrategy (1.0.1 이후)

Anyframe의 TableIdGenService에서는 org.anyframe.idgen.impl.strategy.TimestampStrategy라는 generation strategy를 제공한다. TimestampStrategy는 id generation strategy의 한 종류로, id 생성 시 TableIdGenService 실행을 통해 추출된 순번과 javax.text.SimpleDateFormat pattern으로 생성된 현재 시간을 조합한다. 예를 들어 pattern이 'yyyy-MM-dd', TableIdGenService를 통해 전달된 순번이 '12', separator가 '-', fillChar가 '0', cipers가 5일 경우 전달된 id의 값은 현재 시간과 순번을 조합하여 '2011-05-24-00012'와 같은 형태가 된다. 이 때, 자릿수(cipers)가 5, 순번이 '12'이면 순번은 5자리로 구성하되 빈 부분은 fillChar로 정의된 문자열로 채우기 때문에 '00012'와 같은 형태가 된다. 또한 separator는 현재 시간과 순번을 구분하기 위한 구분자 역할을 한다. pattern과 separator가 입력되지 않았을 경우 기본적으로 'yyyyMMdd', '-'과 같은 값이 적용되어 신규 id가 조합됨을 알아 두도록 한다.

```
<bean name="mixPrefix" class="org.anyframe.idgen.impl.strategy.TimestampStrategy">
  <property name="pattern" value="yyyy-MM-dd"/>
```

```
<property name="separator" value="-"/>
<property name="ciphers" value="5"/>
<property name="fillChar" value="0"/>
</bean>
```

## 4.3.Id Generation Strategy를 implements하는 방법

strategy를 개발하기 위해서는 org.anyframe.idgen.IdGenerationStrategy를 implements 해야한다. 또한 makeId(String originalId) 메소드를 통해 특정 Strategy가 적용된 id를 return하도록 한다. MixPrefixStrategy안의 makeId의 구현 예는 다음과 같다.

```
public class MixPrefixStrategy implements IdGenerationStrategy{
    public String makeId(String originalId) {
        return prefix + StringUtil.fillString(originalId, fillChar, ciphers);
    }
}
```