Id Generation Service

Summary

This service has been designed to createa unique ID required when the system is developed.

Main Concept

Universally Unique Identifier(UUID)

UUID is the standards for identifier established by OSF(Open Software Foundation). UUID consists of 16-byte (128-bit) numbers. There is no specific regulation for the method of expressing UUID, but in general, it is expressed in hexadecimal number such as 8-4-4-4-12.

550e8400-e29b-41d4-a716-446655440000

The UUID standard is described in the following documents.

- ISO/IEC 11578:1996 "Information technology -- Open Systems Interconnection -- Remote Procedure Call (RPC)"
- ITU-T Rec. X.667 | ISO/IEC 9834-8:2005
- IETF Proposed Standard RFC 4122

UUID exists in five versions as below.

- Version 1 (MAC Address)
  : This creates UUID using time information and MAC address of computer creating UUID. Since it uses MAC address of computer, it leaves information regarding at which computer it was created.

- Version 2 (DCE Security)
  : This creates UUID using POSIX UID.

- Version 3 (MD5 Hash)
  : This creates UUID using MD5 from URL.

- Version 4 (Random)
  : Create UUID using Random Number.

- Version 5 (SHA-1 Hash)
  : This creates UUID using SHA-1 Hashing.

Description

As for the type of ID creation, there are three types such as UUID Generation Service to create UUID; Sequence Id Generation Service to utilize sequence and Table Id Generation Service creating by designating the table for key provision.

UUID Generation Service

This creates 16 byte long ID using the UUID creation algorithm in order to create new ID. It supports the ID creation of 2 types such as ID creation of String type and ID creation of BigDecimal. There are 3 kinds of support methods such as Depending on the Mac Address Base Service , IP Address Base Service , No Address Base Servicedepending on setting.

Mac Address Base Service

UUIdGenerationService to create unique Id based on MAC Address.

Configuration

```
<bean name="UUIdGenerationService"
class="egovframework.rte.fdl.idgnr.impl.EgovUUIdGnrService">
<property name="address">
<value>00:00:F0:79:19:5B</value>
</property>
</bean>
```

Sample Source

```
@Resource(name="UUIdGenerationService")
privateEgovIdGnrServiceuUidGenerationService;

@Test
public void testUUIdGeneration() throws Exception {
assertNotNull(uUidGenerationService.getNextStringId());
assertNotNull(uUidGenerationService.getNextBigDecimalId());
}
```

IP Address Base Service

UUIdGenerationService to create unique Id based on IP Address

Configuration

```
<bean name="UUIdGenerationServiceWithIP"
class="egovframework.rte.fdl.idgnr.impl.EgovUUIdGnrService">
<property name="address">
<value>100.128.120.107</value>
</property>
</bean>
```

Sample Source

```
@Resource(name="UUIdGenerationServiceWithIP")
privateEgovIdGnrServiceuUIdGenerationServiceWithIP;

@Test
public void testUUIdGenerationIP() throws Exception {
assertNotNull(uUIdGenerationServiceWithIP.getNextStringId());
assertNotNull(uUIdGenerationServiceWithIP.getNextBigDecimalId());
}
```

No Address Base Service

UUIdGenerationService to create unique ID and to create address information using Math.random() without IP Address setting.

Configuration

```
<bean name="UUIdGenerationServiceWithoutAddress"
class="egovframework.rte.fdl.idgnr.impl.EgovUUIdGnrService">
</bean>
```

Sample Source

```
@Resource(name="UUIdGenerationServiceWithoutAddress")
privateEgovIdGnrServiceuUIdGenerationServiceWithoutAddress;

@Test
public void testUUIdGenerationNoAddress() throws Exception {
assertNotNull(uUIdGenerationServiceWithoutAddress.getNextStringId());
```

```
assertNotNull(uUIdGenerationServiceWithoutAddress.getNextBigDecimalId());
}
```

View Source

Sequence Id Generation Service

This skill uses SEQUENCE of database to create new ID. It enables to create the ID by designating Query at the system using the service. It supports two types such as Basic Type Service andBigDecimal Type Service.

Basic Type Service

As a service to provide basic type ID. It provides int,short,byte,long type of IDs.

DB Schema

```
CREATE SEQUENCE idstest MINVALUE 0;
```

Configuration

```
<bean name="primaryTypeSequenceIds"
class="egovframework.rte.fdl.idgnr.impl.EgovSequenceIdGnrService"
destroy-method="destroy">
<property name="dataSource" ref="dataSource"/>
<property name="query"   value="SELECT idstest.NEXTVAL FROM DUAL"/>
</bean>
```

Sample Source

```
@Resource(name="primaryTypeSequenceIds")
privateEgovIdGnrServiceprimaryTypeSequenceIds;

@Test
public void testPrimaryTypeIdGeneration() throws Exception {
    //int
assertNotNull(primaryTypeSequenceIds.getNextIntegerId());
    //short
assertNotNull(primaryTypeSequenceIds.getNextShortId());
    //byte
assertNotNull(primaryTypeSequenceIds.getNextByteId());
    //long
assertNotNull(primaryTypeSequenceIds.getNextLongId());
}
```

BigDecimal Type Service

As a service to provide BigDecimal ID, it additionally sets useBigDecimals to “true in the Default Type ID Provision Service to use BigDecimal.

DB Schema

```
CREATE SEQUENCE idstest MINVALUE 0;
```

Configuration

```
<bean name="bigDecimalTypeSequenceIds"
class="egovframework.rte.fdl.idgnr.impl.EgovSequenceIdGnrService"
destroy-method="destroy">
<property name="dataSource" ref="dataSource"/>
```

```
<property name="query"   value="SELECT idstest.NEXTVAL FROM DUAL"/>
<property name="useBigDecimals" value="true"/>
</bean>
```

Sample Source

```
@Resource(name="bigDecimalTypeSequenceIds")
privateEgovIdGnrServicebigDecimalTypeSequenceIds;

@Test
public void testBigDecimalTypeIdGeneration() throws Exception {
    //BigDecimal
assertNotNull(bigDecimalTypeSequenceIds.getNextBigDecimalId());
}
```

Setting per Database

Oracle

- DB Schema :
  CREATE SEQUENCE <sequence name> [START WITH <start value>] [INCREMENT BY <increment value>] [MINVALUE <min value>] [MAXVALUE <max value>]
- Query :
  SELECT <sequence name>.NEXTVAL FROM DUAL

HSQL

- DB Schema :
  CREATE SEQUENCE <sequence name> [AS {INTEGER | BIGINT}] [START WITH <start value>] [INCREMENT BY <increment value>]
- Query :
  SELECT NEXT VALUE FOR <sequence name> FROM DUAL
  (Since HSQL DB does not provide DUAL table, it should manually create DUAL table with one row.)

IBM DB2

- DB Schema :
  CREATE SEQUENCE <sequence name> [START WITH <start value>] [INCREMENT BY <increment value>] [MINVALUE <min value>] [MAXVALUE <max value>]
- Query :
  SELECT NEXT VALUE FOR <sequence name> FROM SYSIBM.SYSDUMMY1

Sybase

- Sybase does not support Sequence.

MS SQL Server

- MS SQL Server does not support Sequence.

MySQL

- MySQL does not support Sequence.

Table Id Generation Service

It is the service provided by entering and managing the key value and ID value corresponding to key value in order to get new idea, create separate table. It requires 2 columns such as table_name(CHAR or VARCHAR type), next_id(integer or DECIMAL type).It provides Basic Service that is available using

the information only set in the separate table and Strategy Base Service that can create String ID by designating the prefix and character strings to be filled.

Basic Service

This is a skill to create ID with the information designated in the table. It can be used by creating the table in the system.

DB Schema

```
CREATE TABLE ids ( table_namevarchar(16) NOT NULL,
                    next_id DECIMAL(30) NOT NULL,
                       PRIMARY KEY (table_name));
INSERT INTO ids VALUES('id','0');
```

- The DB Schema information to be created in advance at the system to use ID generation service.

Configuration

```
<bean name="basicService" class="egovframework.rte.fdl.idgnr.impl.EgovTableIdGnrService"
destroy-method="destroy">
<property name="dataSource" ref="dataSource"/>
<property name="blockSize"   value="10"/>
<property name="table"      value="ids"/>
<property name="tableName"   value="id"/>
</bean>
```

- blockSize: an information internally used for Id Generation. Information not to access DB at every request of ID(DB connection processing at every number of times designated.)
- table: being able to change the table name used at the place of use with the created table information
- tableName: thekey value for identifying ID to use (designated as tableName since ID is required per table separately)

Sample Source

```
@Resource(name="basicService")
privateEgovIdGnrServicebasicService;

@Test
public void testBasicService() throws Exception {
    //int
assertNotNull(basicService.getNextIntegerId());
    //short
assertNotNull(basicService.getNextShortId());
    //byte
assertNotNull(basicService.getNextByteId());
    //long
assertNotNull(basicService.getNextLongId());
    //BigDecimal
assertNotNull(basicService.getNextBigDecimalId());
    //String
assertNotNull(basicService.getNextStringId());
}
```

Strategy Base Service

This service registers the rule for creating ID and to create the ID in accordance with rule. It can be used by adding the strategy information setting in the above Basic Service. Provided, however, that this service provides string type ID only.

DB Schema

```
CREATE TABLE idttest(table_namevarchar(16) NOT NULL,
                     next_id DECIMAL(30) NOT NULL,
                        PRIMARY KEY (table_name));
INSERT INTO idttestVALUES('test','0');
```

- The DB Schema information to be created in advance at the system to use ID Generation service.

Configuration

```
<bean name="Ids-TestWithGenerationStrategy"
class="egovframework.rte.fdl.idgnr.impl.EgovTableIdGnrService"
destroy-method="destroy">
<property name="dataSource" ref="dataSource"/>
<property name="strategy" ref="strategy"/>
<property name="blockSize" value="1"/>
<property name="table"     value="idttest"/>
<property name="tableName" value="test"/>
</bean>

<bean name="strategy" class="egovframework.rte.fdl.idgnr.impl.strategy.EgovIdGnrStrategyImpl">
<property name="prefix" value="TEST-"/>
<property name="cipers" value="5"/>
<property name="fillChar" value="*"/>
</bean>
```

- strategy: bean name setting of MixPrefix defined below
- prefix: designating the configuration value to fix and attach in front of ID
- cipers: designating the length of ID except prefix
- fillChar: characters expressed instead of 0

Sample Source

```
@Resource(name="Ids-TestWithGenerationStrategy")
privateEgovIdGnrServiceidsTestWithGenerationStrategy;

@Test
public void testIdGenStrategy() throws Exception {

initializeNextLongId("test", 1);

    // prefix : TEST-, cipers : 5, fillChar :*)
for (inti = 0; i< 5; i++)
assertEquals("TEST-****" + (i + 1), idsTestWithGenerationStrategy.getNextStringId());
}
```

Reference

N/A