Software Documentation : Ruby Custom Acts As Id Generator

This page last changed on Nov 11, 2008 by ivern.

1. Scope

1.1 Overview

While Ruby on Rails provides a built-in mechanism to generate ids for primary keys, TopCoder's internal systems use table-driven id sequences through the Java Id Generator component. The purpose of this component is to provide a Rails plugin that will allow ActiveRecord models to easily generate their primary key ids from legacy Id Generator sequences.

1.1.1 Version

1.0.0

1.2 Logic Requirements

The plugin implements a subset of the functionality provided by the Java Id Generator component. Refer to that component's specifications for details on how to implement particular functionality.

1.2.1 Id Generation

The plugin must add an acts_as_id_generator method to ActiveRecord::Base. When invoked in a model, this method overrides Rails' standard id generation mechanism so that newly created instances of the model use custom ids for their primary key values generated from Id Generator sequence tables.

1.2.2 Exhausted Flag

The plugin must not only respect this sequence flag (it is an error to try to generate an id from an exhausted sequence), but also set it if appropriate.

1.2.3 Concurrency

In order to peacefully coexist with the Java version of Id Generator, the plugin must use pessimistic locking on the appropriate sequence row.

1.2.4 Error Handling

The plugin must correctly handle any errors in id generation (missing table, missing or exhausted sequence, table locks, etc.) Throwing descriptive exceptions on any errors that the plugin can't recover from (missing table or sequence, exhausted sequence) is fine.

1.3 Required Algorithms

The High/Low algorithm for id generation is described in the documentation and implementation of the Java Id Generator component. This plugin must be fully compatible with Id Generator's sequence table implementation, and be able to coexist with it.

1.4 Example of the Software Usage

Regular table and key column names, custom id sequence:

```
class Foo < ActiveRecord::Base
acts_as_id_generator :sequence => 'FOOS_SEQ'
end
```

Custom table and key column names, custom id sequence, custom id sequence table:

```
class Bar < ActiveRecord::Base
set_table_name 'bartastic'
set_primary_key 'bar_id'

acts_as_id_generator :sequence => 'bars_seq', :table => 'my_id_sequences'
end
```

1.5 Future Component Direction

The component may be extended to allow distribution as a Ruby Gem rather than using the Rails plugin system.

2. Interface Requirements

2.1.1 Graphical User Interface Requirements

None.

2.1.2 External Interfaces

The `acts_as_id_generator` method must allow two optional parameters:

- `:sequence` Sequence name to generate ids from. The default value is the table name suffixed with "_seq". For example, "foos_seq".
- `:table` Table to store the id sequences. The default value is "id_sequences".

The sequence table can be created using the following DDL:

```
CREATE TABLE id_sequences (
name VARCHAR(255) PRIMARY KEY,
next_block_start DECIMAL(12,0) NOT NULL,
block_size DECIMAL(10,0) NOT NULL,
exhausted DECIMAL(1,0) NOT NULL DEFAULT 0
);
CREATE UNIQUE INDEX 166_166 ON id_sequences(name);
```

2.1.3 Environment Requirements

- Development language: Ruby 1.8
- Compile target: JRuby 1.1.5

2.1.4 Package Structure

TopCoder::Acts::IdGenerator

3. Software Requirements

3.1 Administration Requirements

3.1.1 What elements of the application need to be configurable?

None.

3.2 Technical Constraints

3.2.1 Are there particular frameworks or standards that are required?

- Ruby on Rails 2.1.2

3.2.2 TopCoder Software Component Dependencies:

None.

3.2.3 Third Party Component, Library, or Product Dependencies:

See QA Environment.

3.2.4 QA Environment:

- JRuby 1.1.5
- Rails 2.1.2
- ActiveRecord JDBC Adapter with Informix support (http://github.com/ivern/activerecord-jdbc-adapter/tree/master)
- Informix 10
- Informix JDBC Adapter 3.00.JC3
- Sun JDK 1.6.0_10

3.3 Design Constraints

The component design and development solutions must adhere to the guidelines as outlined in the TopCoder Software Component Guidelines. Modifications to these guidelines for this component should be detailed below.

3.4 Required Documentation

3.4.1 Design Documentation

- Use-Case Diagram
- Class Diagram
- Sequence Diagram
- Component Specification

3.4.2 Help / User Documentation

- Design documents must clearly define intended component usage in the 'Documentation' tab of Poseidon.