org.jax.mgi.shr.sva Design Document

Author: M Walker

Created: June 9, 2005

Last Modified: June 10, 2005 13:04

1 Purpose of Document

This document will describe the purpose of the org.jax.mgi.shr.sva package from the lib_java_core product. It will provide an overview of the classes used and provide some examples on how to utilize this framework within an application.

2 Introduction

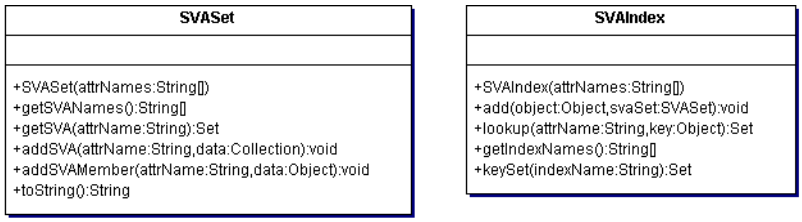
This package is designed for representing and manipulating set valued attributes (SVA), which, as the name implies, are attributes that contain sets as values. No restrictions are imposed by this package on what kinds of objects are represented within the sets. Any collection of objects can be part of an SVA. An SVA can be thought of as a generic Java Map class where the name of the SVA is the key to the Map and the value is a set. The two classes from this package are **SVASet** and **SVAIndex**. **SVASet** provides a container class for storing named SVAs. The **SVASet** is similar in concept to another data structure used within lib_java_core called the **DTO** class (see the org.jax.mgi.shr.dto package). The difference is that a **DTO** has attributes which are single-valued, whereas the attributes of the **SVASet** are multi-valued.

3 Class Overview

The **SVASet** class is a simple class that contains a set of SVAs and methods for accessing and manipulating them. Each of these methods require the name of the target SVA as a parameter. There are methods available to add new values to a named SVA and to access the values of a named SVA.

The **SVAIndex** class is a class which indexes objects by the values from the **SVASets** they are associated with. If an object and an **SVASet** are provided to the **SVAIndex** then the object is internally stored with indexes pointing to from each of the **SVASet** values. Internally, there are hashtables being used for each named SVA from the **SVASets**. These hashtables are named after the SVA names and they provide a way to lookup associated objects by a the values of an SVA.

4 Class Diagram



5 Application Level Usage

Any object can store attributes within an **SVASet** by instantiating a new instance and calling the add() methods. The constructor takes an array of Strings. This array represent the list of SVA names that this instance will manage. Any calls to add() methods specifying a name outside of this list is treated as a noopt call and no action is taken and no exception is raised by the method call. Two methods are available for adding values, addSVA(String, Collection) and addSVAMember(String, Object). The addSVAMember() method takes the name of the SVA and the object to add. This object is simply added to the set for the named SVA. The addSVA() takes the name of the SVA and a Collection of objects. These objects are merged in to the set of values for the named SVA. The method getSVA(String) takes the name of an SVA supported by this instance and returns the corresponding set for the named SVA. The method getSVANames() return an array of Strings which contains all the SVA names supported by this instance.

Once an **SVASet** is created the **SVAIndex** provides a way to associate an object with an **SVASet** and provides lookup methods for finding these objects by the attributes of the associated **SVASet**. For example, if your **SVASet** represents a gene with associated sequences from various sequence providers, it would contains SVAs named after the providers and each of these SVAs would contain a set of sequences associated to the gene from the corresponding provider. The **SVAIndex** class could be used to associate an **SVASet** to a gene object and provide indexing on these gene objects by their associated sequences.