org.jax.mgi.shr.dto Design Document

Author: jsb

Created: June 10, 2003

Last Modified: May 21, 2004 13:08

1 Purpose of Document

This document outlines the design for the `org.jax.mgi.shr.dto` package.

2 Organizing Principle

All classes contained in `org.jax.mgi.shr.dto` deal with Data Transfer Objects (DTO). A DTO is used to transfer data easily between various pieces of a system, such as between servlets and JSPs. They can be thought of as generic beans, allowing one to set and get many different values.

3 Configuration

All classes in `org.jax.mgi.shr.dto` are stand-alone; no configuration is necessary.

4 Classes

Rather than present a detailed API for each class, we only present an overview here. The details are covered in the javadocs for each class, so we do not replicate them here.

4.1 DTO

A `DTO` represents an association of names and values. The values may also be complex objects, including other `DTO`s. The `DTO` class implements the `java.util.Map` interface, leading to good integration with the standard Java classes. A `DTO`'s public methods facilitate such things as: setting a value, getting a value, getting a String representation of a value, getting a list of names defined in the `DTO`, merging one `DTO` into another, and resetting the `DTO` to be empty.

The `DTO` class does not provide a public constructor. Instead, the class maintains a pool of available `DTO`s. This should increase efficiency by decreasing the number of objects that need to be created, especially for long-running systems. There are two class methods: one to get an empty `DTO` and one to return a `DTO` to the pool. If the pool is empty, then the "`getDTO()`" method will instantiate a new one. If the pool is full, then the "`putDTO()`" method simply lets the returned one be garbage collected.

4.2 DTOConstants

The `DTOConstants` class simply defines a set of constants for use when adding values to `DTO`s. It is possible to use whatever Strings you want for names, but this set provides constants for some commonly used ones. Using these constants should help minimize future code changes.

Example 1: acquiring a DTO, adding fields to it, and releasing it

```
// get a DTO from the pool of available ones
DTO myDTO = DTO.getDTO();

// add stuff to the DTO and do any other processing
myDTO.set (DTOConstants.MarkerSymbol, "Kit");
myDTO.set (DTOConstants.MarkerKey, new Integer (10603));
...

// clear this DTO and return it to the pool of available ones
DTO.putDTO (myDTO);
```