WorkingWithStaticDataObjects

WorkingWithStaticDataObjects

Although the RDBDAS default behavior is to produce graphs of dynamic DataObjects (see ResultSetDrivenDataObjectTypes), the DAS allows users to work with Static DataObjects as well.

Clients can designate a set of Types by providing a piece of configuration (usually in the form of a configuration XML file) to the DAS and the DAS will then use these Types for the DataObject model.

Without any name mapping configuration specified, the DAS will attempt to use DataObject Type and Property names equivalent to database Table and Column names. For example, if the database table name is "BANK_EMPLOYEE", the DAS will attempt to create DataObject instances of type BANK_EMPLOYEE. If this Type doesn't exist in the provided Types, an exception will be thrown by SDO.

To map these database tables and columns to your SDO types and properties, you can provide a "Type Name" and "Property Name" (see WorkingWithNameMapping). In the above example, if we specify a Type name of "employee", the DAS will attempt to create DataObject instances of type "employee" from the information in the BANK_EMPLOYEE table.

It is not necessary to specify any sort of mapping between database column data types and your Property data types. If the data types do not match, a conversion will be made between the two when the property is set on the SDO model. The possible data type conversions are specified in the SDO specification. If there is no valid conversion available between the two data types, you can specify a Converter (see WorkingWithColumnConverters) that the DAS will use to convert the data before setting the value in the DataObject.

The following example demostrates the RDB DAS wih Static SDO Types:

```
SDOUtil.registerStaticTypes(CustomerFactory.class);

DAS das = DAS.FACTORY.createDAS(getConfig("staticCustomerOrder.xml"), getConnection());
Command select = das.getCommand("Customer and Orders");
select.setParameter(1, Integer.valueOf(1));
DataObject root = select.executeQuery();

// Modify a customer
Customer customer = (Customer) root.getDataObject("Customer[1]");
customer.setLastName("Pavick");

// Modify an order
AnOrder order = (AnOrder) customer.getOrders().get(0);
order.setProduct("Kitchen Sink 001");

// Flush changes
das.applyChanges((DataObject) root);
```

And, here is the associated config file:

```
<Config xsi:noNamespaceSchemaLocation="http:///org.apache.tuscany.das.rdb/config.xsd" xmlns:xsi="http://www.w3.
org/2001/XMLSchema-instance"
     dataObjectModel="http:///org.apache.tuscany.das.rdb.test/customer.xsd">

    <Command name="Customer and Orders" SQL="SELECT * FROM CUSTOMER LEFT JOIN ANORDER ON CUSTOMER.ID = ANORDER.
CUSTOMER_ID where CUSTOMER.ID = ?" kind="Select"/>

    <Table tableName="CUSTOMER" typeName="Customer">
      <Column columnName="ID" primaryKey="true"/>
    </Table>

    <Table tableName="ANORDER" typeName="AnOrder">
      <Column columnName="ID" primaryKey="true"/>
      <Column columnName="CUSTOMER_ID"/>
    </Table>

    <Relationship name="orders" primaryKeyTable="CUSTOMER" foreignKeyTable="ANORDER" many="true">
       <KeyPair primaryKeyColumn="ID" foreignKeyColumn="CUSTOMER_ID"/>
    </Relationship>

   </Config>
```

The first line of the example registers the Static model with the SDO runtime. This step would not normally be intermixed with the data access code but the Types must be registered by some part of the applicaiton. The DAS is provided with the URI or the static types via the config file via the dataObjectModel attribute.

You can see that the sample makes use of the Types provided.