Dynamic DAO Implementation

Example dynamic-dao-implementation can be browsed at https://github.com/apache/tomee/tree/master/examples/dynamic-dao-implementation

Many aspects of Data Access Objects (DAOs) are very repetitive and boiler plate. As a fun and experimental feature, TomEE supports dynamically implementing an interface that is seen to have standard DAO-style methods.

The interface has to be annotated with @PersistenceContext to define which EntityManager to use.

Methods should respect these conventions:

- void save(Foo foo): persist foo
- Foo update(Foo foo): merge foo
- void delete(Foo foo): remove foo, if foo is detached it tries to attach it
- Collection<Foo>|Foo namedQuery(String name[, Map<String, ?> params, int first, int max]): run the named query called name, params contains bindings, first and max are used for magination. Last three parameters are optionnals
- Collection<Foo>|Foo nativeQuery(String name[, Map<String, ?> params, int first, int max]): run the native query called name, params contains bindings, first and max are used for magination. Last three parameters are optionnals
- Collection<Foo>|Foo query(String value [, Map<String, ?> params, int first, int max]): run the query put as first parameter, params contains bindings, first and max are used for magination. Last three parameters are optionnals
- Collection<Foo> findAll([int first, int max]): find all Foo, parameters are used for pagination
- Collection<Foo> findByBar1AndBar2AndBar3(<bar 1 type> bar1, <bar 2 type> bar2, <bar3 type> bar3 [, int first, int max]): find all Foo with specified field values for bar1, bar2, bar3.

Dynamic finder can have as much as you want field constraints. For String like is used and for other type equals is used.

Example

User

```
package org.superbiz.dynamic;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;

@Entity
@NamedQueries({
        @NamedQuery(name = "dynamic-ejb-impl-test.query", query = "SELECT u FROM User
AS u WHERE u.name LIKE :name"),
        @NamedQuery(name = "dynamic-ejb-impl-test.all", query = "SELECT u FROM User AS
u")
})
public class User {
    @Id
    @GeneratedValue
    private long id;
    private String name;
    private int age;

    public long getId() {
        return id;
    }

    public void setId(long id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }
}
```

UserDao

```java
package org.superbiz.dynamic;


import javax.ejb.Stateless;
import javax.persistence.PersistenceContext;
import java.util.Collection;
import java.util.Map;

@Stateless
@PersistenceContext(name = "dynamic")
public interface UserDao {
    User findById(long id);

    Collection<User> findByName(String name);

    Collection<User> findByNameAndAge(String name, int age);

    Collection<User> findAll();

    Collection<User> findAll(int first, int max);

    Collection<User> namedQuery(String name, Map<String, ?> params, int first, int
max);

    Collection<User> namedQuery(String name, int first, int max, Map<String, ?>
params);

    Collection<User> namedQuery(String name, Map<String, ?> params);

    Collection<User> namedQuery(String name);

    Collection<User> query(String value, Map<String, ?> params);

    void save(User u);

    void delete(User u);

    User update(User u);
}
```

persistence.xml

```
<persistence version="2.0"
             xmlns="http://java.sun.com/xml/ns/persistence"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:schemaLocation="
      http://java.sun.com/xml/ns/persistence
      http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd">
  <persistence-unit name="dynamic" transaction-type="JTA">
    <jta-data-source>jdbc/dynamicDB</jta-data-source>
    <class>org.superbiz.dynamic.User</class>
    <properties>
      <property name="openjpa.jdbc.SynchronizeMappings" value=
"buildSchema(ForeignKeys=true)"/>
    </properties>
  </persistence-unit>
</persistence>
```

DynamicUserDaoTest

```
package org.superbiz.dynamic;

import junit.framework.Assert;
import org.junit.BeforeClass;
import org.junit.Test;

import javax.ejb.EJBException;
import javax.ejb.Stateless;
import javax.ejb.embeddable.EJBContainer;
import javax.naming.Context;
import javax.persistence.EntityManager;
import javax.persistence.NoResultException;
import javax.persistence.PersistenceContext;
import java.util.Collection;
import java.util.HashMap;
import java.util.Map;
import java.util.Properties;

import static junit.framework.Assert.assertEquals;
import static junit.framework.Assert.assertNotNull;
import static junit.framework.Assert.assertTrue;

public class DynamicUserDaoTest {
    private static UserDao dao;
    private static Util util;

    @BeforeClass
    public static void init() throws Exception {
        final Properties p = new Properties();
        p.put("jdbc/dynamicDB", "new://Resource?type=DataSource");
        p.put("jdbc/dynamicDB.JdbcDriver", "org.hsqldb.jdbcDriver");
        p.put("jdbc/dynamicDB.JdbcUrl", "jdbc:hsqldb:mem:moviedb");
        p.put("jdbc/dynamicDB.UserName", "sa");
        p.put("jdbc/dynamicDB.Password", "");

        final Context context = EJBContainer.createEJBContainer(p).getContext();
        dao = (UserDao) context.lookup("java:global/dynamic-dao-
implementation/UserDao");
        util = (Util) context.lookup("java:global/dynamic-dao-implementation/Util");

        util.init(); // init database
    }

    @Test
    public void simple() {
        User user = dao.findById(1);
        assertNotNull(user);
        assertEquals(1, user.getId());
    }

    @Test
    public void findAll() {
        Collection<User> users = dao.findAll();
        assertEquals(10, users.size());
    }

    @Test
    public void pagination() {
        Collection<User> users = dao.findAll(0, 5);
        assertEquals(5, users.size());

        users = dao.findAll(6, 1);
        assertEquals(1, users.size());
        assertEquals(7, users.iterator().next().getId());
    }

    @Test
    public void persist() {
        User u = new User();
        dao.save(u);
        assertNotNull(u.getId());
        util.remove(u);
    }

    @Test
    public void remove() {
        User u = new User();
        dao.save(u);
        assertNotNull(u.getId());
        dao.delete(u);
        try {
            dao.findById(u.getId());
            Assert.fail();
        } catch (EJBException ee) {
            assertTrue(ee.getCause() instanceof NoResultException);
        }
    }

    @Test
    public void merge() {
        User u = new User();
        u.setAge(1);
        dao.save(u);
        assertEquals(1, u.getAge());
        assertNotNull(u.getId());

        u.setAge(2);
        dao.update(u);
        assertEquals(2, u.getAge());

        dao.delete(u);
    }

    @Test
    public void oneCriteria() {
        Collection<User> users = dao.findByName("foo");
        assertEquals(4, users.size());
        for (User user : users) {
            assertEquals("foo", user.getName());
        }
    }

    @Test
    public void twoCriteria() {
        Collection<User> users = dao.findByNameAndAge("bar-1", 1);
        assertEquals(1, users.size());

        User user = users.iterator().next();
        assertEquals("bar-1", user.getName());
        assertEquals(1, user.getAge());
    }

    @Test
    public void query() {
        Map<String, Object> params = new HashMap<String, Object>();
        params.put("name", "foo");

        Collection<User> users = dao.namedQuery("dynamic-ejb-impl-test.query", params,
0, 100);
        assertEquals(4, users.size());

        users = dao.namedQuery("dynamic-ejb-impl-test.query", params);
```

```java
        assertEquals(4, users.size());

        users = dao.namedQuery("dynamic-ejb-impl-test.query", params, 0, 2);
        assertEquals(2, users.size());

        users = dao.namedQuery("dynamic-ejb-impl-test.query", 0, 2, params);
        assertEquals(2, users.size());

        users = dao.namedQuery("dynamic-ejb-impl-test.all");
        assertEquals(10, users.size());

        params.remove("name");
        params.put("age", 1);
        users = dao.query("SELECT u FROM User AS u WHERE u.age = :age", params);
        assertEquals(3, users.size());
    }

    @Stateless
    public static class Util {
        @PersistenceContext
        private EntityManager em;

        public void remove(User o) {
            em.remove(em.find(User.class, o.getId()));
        }

        public void init() {
            for (int i = 0; i < 10; i++) {
                User u = new User();
                u.setAge(i % 4);
                if (i % 3 == 0) {
                    u.setName("foo");
                } else {
                    u.setName("bar-" + i);
                }
                em.persist(u);
            }
        }
    }
}
```

Running

```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running org.superbiz.dynamic.DynamicUserDaoTest
Apache OpenEJB 4.0.0-beta-1    build: 20111002-04:06
```

```
http://tomee.apache.org/
INFO - openejb.home = /Users/dblevins/examples/dynamic-dao-implementation
INFO - openejb.base = /Users/dblevins/examples/dynamic-dao-implementation
INFO - Using 'javax.ejb.embeddable.EJBContainer=true'
INFO - Configuring Service(id=Default Security Service, type=SecurityService,
provider-id=Default Security Service)
INFO - Configuring Service(id=Default Transaction Manager, type=TransactionManager,
provider-id=Default Transaction Manager)
INFO - Configuring Service(id=jdbc/dynamicDB, type=Resource, provider-id=Default JDBC
Database)
INFO - Found EjbModule in classpath: /Users/dblevins/examples/dynamic-dao-
implementation/target/classes
INFO - Found EjbModule in classpath: /Users/dblevins/examples/dynamic-dao-
implementation/target/test-classes
INFO - Beginning load: /Users/dblevins/examples/dynamic-dao-
implementation/target/classes
INFO - Beginning load: /Users/dblevins/examples/dynamic-dao-
implementation/target/test-classes
INFO - Configuring enterprise application: /Users/dblevins/examples/dynamic-dao-
implementation
INFO - Configuring Service(id=Default Stateless Container, type=Container, provider-
id=Default Stateless Container)
INFO - Auto-creating a container for bean UserDao: Container(type=STATELESS,
id=Default Stateless Container)
INFO - Configuring Service(id=Default Managed Container, type=Container, provider-
id=Default Managed Container)
INFO - Auto-creating a container for bean org.superbiz.dynamic.DynamicUserDaoTest:
Container(type=MANAGED, id=Default Managed Container)
INFO - Configuring PersistenceUnit(name=dynamic)
INFO - Auto-creating a Resource with id 'jdbc/dynamicDBNonJta' of type 'DataSource for
'dynamic'.
INFO - Configuring Service(id=jdbc/dynamicDBNonJta, type=Resource, provider-
id=jdbc/dynamicDB)
INFO - Adjusting PersistenceUnit dynamic <non-jta-data-source> to Resource ID
'jdbc/dynamicDBNonJta' from 'null'
INFO - Enterprise application "/Users/dblevins/examples/dynamic-dao-implementation"
loaded.
INFO - Assembling app: /Users/dblevins/examples/dynamic-dao-implementation
INFO - PersistenceUnit(name=dynamic,
provider=org.apache.openjpa.persistence.PersistenceProviderImpl) - provider time 417ms
INFO - Jndi(name="java:global/dynamic-dao-
implementation/UserDao!org.superbiz.dynamic.UserDao")
INFO - Jndi(name="java:global/dynamic-dao-implementation/UserDao")
INFO - Jndi(name="java:global/dynamic-dao-
implementation/Util!org.superbiz.dynamic.DynamicUserDaoTest$Util")
INFO - Jndi(name="java:global/dynamic-dao-implementation/Util")
INFO -
Jndi(name="java:global/EjbModule346613126/org.superbiz.dynamic.DynamicUserDaoTest!org.
superbiz.dynamic.DynamicUserDaoTest")
INFO -
Jndi(name="java:global/EjbModule346613126/org.superbiz.dynamic.DynamicUserDaoTest")
```

```
INFO - Created Ejb(deployment-id=UserDao, ejb-name=UserDao, container=Default
Stateless Container)
INFO - Created Ejb(deployment-id=Util, ejb-name=Util, container=Default Stateless
Container)
INFO - Created Ejb(deployment-id=org.superbiz.dynamic.DynamicUserDaoTest, ejb-
name=org.superbiz.dynamic.DynamicUserDaoTest, container=Default Managed Container)
INFO - Started Ejb(deployment-id=UserDao, ejb-name=UserDao, container=Default
Stateless Container)
INFO - Started Ejb(deployment-id=Util, ejb-name=Util, container=Default Stateless
Container)
INFO - Started Ejb(deployment-id=org.superbiz.dynamic.DynamicUserDaoTest, ejb-
name=org.superbiz.dynamic.DynamicUserDaoTest, container=Default Managed Container)
INFO - Deployed Application(path=/Users/dblevins/examples/dynamic-dao-implementation)
WARN - Meta class "org.superbiz.dynamic.User_" for entity class
org.superbiz.dynamic.User can not be registered with following exception
"java.security.PrivilegedActionException: java.lang.ClassNotFoundException:
org.superbiz.dynamic.User_"
WARN - Query "SELECT u FROM User AS u WHERE u.name LIKE :name" is removed from cache
excluded permanently. Query "SELECT u FROM User AS u WHERE u.name LIKE :name" is not
cached because it uses pagination..
Tests run: 9, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 2.471 sec

Results :

Tests run: 9, Failures: 0, Errors: 0, Skipped: 0
```