Dynamic Datasource Routing

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

The TomEE dynamic datasource api aims to allow to use multiple data sources as one from an application point of view.

It can be useful for technical reasons (load balancing for example) or more generally functionnal reasons (filtering, aggregation, enriching…). However please note you can choose only one datasource by transaction. It means the goal of this feature is not to switch more than once of datasource in a transaction. The following code will not work:

```
@Stateless
public class MyEJB {
    @Resource private MyRouter router;
    @PersistenceContext private EntityManager em;

    public void workWithDataSources() {
        router.setDataSource("ds1");
        em.persist(new MyEntity());

        router.setDataSource("ds2"); // same transaction -> this invocation doesn't
work
        em.persist(new MyEntity());
    }
}
```

In this example the implementation simply use a datasource from its name and needs to be set before using any JPA operation in the transaction (to keep the logic simple in the example).

The implementation of the Router

Our router has two configuration parameters: * a list of jndi names representing datasources to use * a default datasource to use

Router implementation

The interface Router (`org.apache.openejb.resource.jdbc.Router`) has only one method to implement, `public DataSource getDataSource()`

Our `DeterminedRouter` implementation uses a ThreadLocal to manage the currently used datasource.

Keep in mind JPA used more than once the getDatasource() method for one operation. To change the datasource in one transaction is dangerous and should be avoid.

```java
package org.superbiz.dynamicdatasourcerouting;

import org.apache.openejb.resource.jdbc.AbstractRouter;

import javax.naming.NamingException;
import javax.sql.DataSource;
import java.util.Map;
import java.util.concurrent.ConcurrentHashMap;

public class DeterminedRouter extends AbstractRouter {
    private String dataSourceNames;
    private String defaultDataSourceName;
    private Map<String, DataSource> dataSources = null;
    private ThreadLocal<DataSource> currentDataSource = new ThreadLocal<DataSource>();

    /**
     * @param datasourceList datasource resource name, separator is a space
     */
    public void setDataSourceNames(String datasourceList) {
        dataSourceNames = datasourceList;
    }

    /**
     * lookup datasource in openejb resources
     */
    private void init() {
        dataSources = new ConcurrentHashMap<String, DataSource>();
        for (String ds : dataSourceNames.split(" ")) {
            try {
                Object o = getOpenEJBResource(ds);
                if (o instanceof DataSource) {
                    dataSources.put(ds, DataSource.class.cast(o));
                }
            } catch (NamingException e) {
                // ignored
            }
        }
    }

    /**
     * @return the user selected data source if it is set
     *         or the default one
     *  @throws IllegalArgumentException if the data source is not found
     */
    @Override
    public DataSource getDataSource() {
        // lazy init of routed datasources
        if (dataSources == null) {
            init();
        }

        // if no datasource is selected use the default one
        if (currentDataSource.get() == null) {
            if (dataSources.containsKey(defaultDataSourceName)) {
                return dataSources.get(defaultDataSourceName);

            } else {
                throw new IllegalArgumentException("you have to specify at least one
datasource");
            }
        }

        // the developper set the datasource to use
        return currentDataSource.get();
    }

    /**
     *
     * @param datasourceName data source name
     */
    public void setDataSource(String datasourceName) {
        if (dataSources == null) {
            init();
        }
        if (!dataSources.containsKey(datasourceName)) {
            throw new IllegalArgumentException("data source called " + datasourceName
+ " can't be found.");
        }
        DataSource ds = dataSources.get(datasourceName);
        currentDataSource.set(ds);
    }

    /**
     * reset the data source
     */
    public void clear() {
        currentDataSource.remove();
    }

    public void setDefaultDataSourceName(String name) {
        this.defaultDataSourceName = name;
    }
}
```

Declaring the implementation

To be able to use your router as a resource you need to provide a service configuration. It is done in a file you can find in META-INF/org.router/ and called service-jar.xml (for your implementation you can of course change the package name).

It contains the following code:

```
<ServiceJar>
  <ServiceProvider id="DeterminedRouter" <!-- the name you want to use -->
      service="Resource"
      type="org.apache.openejb.resource.jdbc.Router"
      class-name="org.superbiz.dynamicdatasourcerouting.DeterminedRouter"> <!--
implementation class -->

    # the parameters

    DataSourceNames
    DefaultDataSourceName
  </ServiceProvider>
</ServiceJar>
```

Using the Router

Here we have a `RoutedPersister` stateless bean which uses our `DeterminedRouter`

```
package org.superbiz.dynamicdatasourcerouting;

import javax.annotation.Resource;
import javax.ejb.Stateless;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;

@Stateless
public class RoutedPersister {
    @PersistenceContext(unitName = "router")
    private EntityManager em;

    @Resource(name = "My Router", type = DeterminedRouter.class)
    private DeterminedRouter router;

    public void persist(int id, String name, String ds) {
        router.setDataSource(ds);
        em.persist(new Person(id, name));
    }
}
```

The test

In test mode and using property style configuration the foolowing configuration is used:

```
public class DynamicDataSourceTest {
    @Test
    public void route() throws Exception {
        String[] databases = new String[]{"database1", "database2", "database3"};

        Properties properties = new Properties();
        properties.setProperty(Context.INITIAL_CONTEXT_FACTORY,
LocalInitialContextFactory.class.getName());

        // resources
        // datasources
        for (int i = 1; i <= databases.length; i++) {
            String dbName = databases[i - 1];
            properties.setProperty(dbName, "new://Resource?type=DataSource");
            dbName += ".";
            properties.setProperty(dbName + "JdbcDriver", "org.hsqldb.jdbcDriver");
            properties.setProperty(dbName + "JdbcUrl", "jdbc:hsqldb:mem:db" + i);
            properties.setProperty(dbName + "UserName", "sa");
            properties.setProperty(dbName + "Password", "");
            properties.setProperty(dbName + "JtaManaged", "true");
        }

        // router
        properties.setProperty("My Router",
"new://Resource?provider=org.router:DeterminedRouter&type=" + DeterminedRouter.class
.getName());
        properties.setProperty("My Router.DatasourceNames", "database1 database2
database3");
        properties.setProperty("My Router.DefaultDataSourceName", "database1");

        // routed datasource
        properties.setProperty("Routed Datasource",
"new://Resource?provider=RoutedDataSource&type=" + Router.class.getName());
        properties.setProperty("Routed Datasource.Router", "My Router");

        Context ctx = EJBContainer.createEJBContainer(properties).getContext();
        RoutedPersister ejb = (RoutedPersister) ctx.lookup("java:global/dynamic-
datasource-routing/RoutedPersister");
        for (int i = 0; i < 18; i++) {
            // persisting a person on database db -> kind of manual round robin
            String name = "record " + i;
            String db = databases[i % 3];
            ejb.persist(i, name, db);
        }
        // assert database records number using jdbc
        for (int i = 1; i <= databases.length; i++) {
            Connection connection = DriverManager.getConnection("jdbc:hsqldb:mem:db" +
i, "sa", "");
            Statement st = connection.createStatement();
            ResultSet rs = st.executeQuery("select count(*) from PERSON");
            rs.next();
            assertEquals(6, rs.getInt(1));
            st.close();
            connection.close();
        }

        ctx.close();
    }
}
```

Configuration via openejb.xml

The testcase above uses properties for configuration. The identical way to do it via the `conf/openejb.xml` is as follows:

```
<!-- Router and datasource -->
<Resource id="My Router" type=
"org.apache.openejb.router.test.DynamicDataSourceTest$DeterminedRouter" provider=
"org.routertest:DeterminedRouter">
    DatasourceNames = database1 database2 database3
    DefaultDataSourceName = database1
</Resource>
```

```
<Resource id="Routed Datasource" type="org.apache.openejb.resource.jdbc.Router"
provider="RoutedDataSource">
    Router = My Router
</Resource>
```

```
<!-- real datasources -->
<Resource id="database1" type="DataSource">
    JdbcDriver = org.hsqldb.jdbcDriver
    JdbcUrl = jdbc:hsqldb:mem:db1
    UserName = sa
    Password
    JtaManaged = true
</Resource>
```

```
<Resource id="database2" type="DataSource">
    JdbcDriver = org.hsqldb.jdbcDriver
    JdbcUrl = jdbc:hsqldb:mem:db2
    UserName = sa
    Password
    JtaManaged = true
</Resource>
```

```
<Resource id="database3" type="DataSource">
    JdbcDriver = org.hsqldb.jdbcDriver
    JdbcUrl = jdbc:hsqldb:mem:db3
    UserName = sa
    Password
    JtaManaged = true
</Resource>
```

Some hack for OpenJPA

Using more than one datasource behind one EntityManager means the databases are already created. If it is not the case, the JPA provider has to create the datasource at boot time.

Hibernate do it so if you declare your databases it will work. However with OpenJPA (the default JPA provider for OpenEJB), the creation is lazy and it happens only once so when you'll switch of database it will no more work.

Of course OpenEJB provides @Singleton and @Startup features of Java EE 6 and we can do a bean just making a simple find, even on none existing entities, just to force the database creation:

```java
@Startup
@Singleton
public class BoostrapUtility {
    // inject all real databases

    @PersistenceContext(unitName = "db1")
    private EntityManager em1;

    @PersistenceContext(unitName = "db2")
    private EntityManager em2;

    @PersistenceContext(unitName = "db3")
    private EntityManager em3;

    // force database creation

    @PostConstruct
    @TransactionAttribute(TransactionAttributeType.SUPPORTS)
    public void initDatabase() {
        em1.find(Person.class, 0);
        em2.find(Person.class, 0);
        em3.find(Person.class, 0);
    }
}
```

Using the routed datasource

Now you configured the way you want to route your JPA operation, you registered the resources and you initialized your databases you can use it and see how it is simple:

```java
@Stateless
public class RoutedPersister {
    // injection of the "proxied" datasource
    @PersistenceContext(unitName = "router")
    private EntityManager em;

    // injection of the router you need it to configured the database
    @Resource(name = "My Router", type = DeterminedRouter.class)
    private DeterminedRouter router;

    public void persist(int id, String name, String ds) {
        router.setDataSource(ds); // configuring the database for the current
transaction
        em.persist(new Person(id, name)); // will use ds database automatically
    }
}
```

Running

```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running org.superbiz.dynamicdatasourcerouting.DynamicDataSourceTest
Apache OpenEJB 4.0.0-beta-1    build: 20111002-04:06
http://tomee.apache.org/
INFO - openejb.home = /Users/dblevins/examples/dynamic-datasource-routing
INFO - openejb.base = /Users/dblevins/examples/dynamic-datasource-routing
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=My Router, type=Resource, provider-id=DeterminedRouter)
INFO - Configuring Service(id=database3, type=Resource, provider-id=Default JDBC
Database)
INFO - Configuring Service(id=database2, type=Resource, provider-id=Default JDBC
Database)
INFO - Configuring Service(id=Routed Datasource, type=Resource, provider-
id=RoutedDataSource)
INFO - Configuring Service(id=database1, type=Resource, provider-id=Default JDBC
Database)
INFO - Found EjbModule in classpath: /Users/dblevins/examples/dynamic-datasource-
routing/target/classes
INFO - Beginning load: /Users/dblevins/examples/dynamic-datasource-
routing/target/classes
INFO - Configuring enterprise application: /Users/dblevins/examples/dynamic-
datasource-routing
WARN - Method 'lookup' is not available for 'javax.annotation.Resource'. Probably
using an older Runtime.
INFO - Configuring Service(id=Default Singleton Container, type=Container, provider-
id=Default Singleton Container)
INFO - Auto-creating a container for bean BoostrapUtility: Container(type=SINGLETON,
id=Default Singleton Container)
INFO - Configuring Service(id=Default Stateless Container, type=Container, provider-
id=Default Stateless Container)
INFO - Auto-creating a container for bean RoutedPersister: Container(type=STATELESS,
id=Default Stateless Container)
INFO - Auto-linking resource-ref 'java:comp/env/My Router' in bean RoutedPersister to
Resource(id=My Router)
INFO - Configuring Service(id=Default Managed Container, type=Container, provider-
id=Default Managed Container)
INFO - Auto-creating a container for bean
org.superbiz.dynamicdatasourcerouting.DynamicDataSourceTest: Container(type=MANAGED,
id=Default Managed Container)
INFO - Configuring PersistenceUnit(name=router)
INFO - Configuring PersistenceUnit(name=db1)
INFO - Auto-creating a Resource with id 'database1NonJta' of type 'DataSource for
'db1'.
INFO - Configuring Service(id=database1NonJta, type=Resource, provider-id=database1)
INFO - Adjusting PersistenceUnit db1 <non-jta-data-source> to Resource ID
'database1NonJta' from 'null'
INFO - Configuring PersistenceUnit(name=db2)
INFO - Auto-creating a Resource with id 'database2NonJta' of type 'DataSource for
'db2'.
INFO - Configuring Service(id=database2NonJta, type=Resource, provider-id=database2)
INFO - Adjusting PersistenceUnit db2 <non-jta-data-source> to Resource ID
'database2NonJta' from 'null'
INFO - Configuring PersistenceUnit(name=db3)
INFO - Auto-creating a Resource with id 'database3NonJta' of type 'DataSource for
'db3'.
INFO - Configuring Service(id=database3NonJta, type=Resource, provider-id=database3)
INFO - Adjusting PersistenceUnit db3 <non-jta-data-source> to Resource ID
'database3NonJta' from 'null'
INFO - Enterprise application "/Users/dblevins/examples/dynamic-datasource-routing"
loaded.
INFO - Assembling app: /Users/dblevins/examples/dynamic-datasource-routing
INFO - PersistenceUnit(name=router,
provider=org.apache.openjpa.persistence.PersistenceProviderImpl) - provider time 504ms
INFO - PersistenceUnit(name=db1,
provider=org.apache.openjpa.persistence.PersistenceProviderImpl) - provider time 11ms
INFO - PersistenceUnit(name=db2,
provider=org.apache.openjpa.persistence.PersistenceProviderImpl) - provider time 7ms
INFO - PersistenceUnit(name=db3,
provider=org.apache.openjpa.persistence.PersistenceProviderImpl) - provider time 6ms
INFO - Jndi(name="java:global/dynamic-datasource-
routing/BoostrapUtility!org.superbiz.dynamicdatasourcerouting.BoostrapUtility")
INFO - Jndi(name="java:global/dynamic-datasource-routing/BoostrapUtility")
INFO - Jndi(name="java:global/dynamic-datasource-
routing/RoutedPersister!org.superbiz.dynamicdatasourcerouting.RoutedPersister")
INFO - Jndi(name="java:global/dynamic-datasource-routing/RoutedPersister")
INFO -
Jndi(name="java:global/EjbModule1519652738/org.superbiz.dynamicdatasourcerouting.Dynam
icDataSourceTest!org.superbiz.dynamicdatasourcerouting.DynamicDataSourceTest")
INFO -
Jndi(name="java:global/EjbModule1519652738/org.superbiz.dynamicdatasourcerouting.Dynam
icDataSourceTest")
INFO - Created Ejb(deployment-id=RoutedPersister, ejb-name=RoutedPersister,
container=Default Stateless Container)
INFO - Created Ejb(deployment-
id=org.superbiz.dynamicdatasourcerouting.DynamicDataSourceTest, ejb-
name=org.superbiz.dynamicdatasourcerouting.DynamicDataSourceTest, container=Default
Managed Container)
INFO - Created Ejb(deployment-id=BoostrapUtility, ejb-name=BoostrapUtility,
container=Default Singleton Container)
INFO - Started Ejb(deployment-id=RoutedPersister, ejb-name=RoutedPersister,
container=Default Stateless Container)
INFO - Started Ejb(deployment-
id=org.superbiz.dynamicdatasourcerouting.DynamicDataSourceTest, ejb-
name=org.superbiz.dynamicdatasourcerouting.DynamicDataSourceTest, container=Default
Managed Container)
INFO - Started Ejb(deployment-id=BoostrapUtility, ejb-name=BoostrapUtility,
container=Default Singleton Container)
INFO - Deployed Application(path=/Users/dblevins/examples/dynamic-datasource-routing)
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 2.504 sec

Results :

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
```