Ojp-delta-token

Support for Delta Token

Overview

Delta Tokens are supported in Olingo JPA Processor Library from Version 1.3.0-SNAPSHOT. Following page explains how JPA applications can generate and handle delta tokens. Please note 1.3.0-SNAPSHOT is yet to be released.

The JPA processor library supports two different approaches for handling delta tokens.

- Implementing methods of abstract class - org.apache.olingo.odata2.jpa.processor.api.ODataJPATombstoneEntityListener
- Implementing no methods of abstract class - org.apache.olingo.odata2.jpa.processor.api.ODataJPATombstoneEntityListener

The class that extends the above class should be registered with the JPA entity as an EntityListener as defined in JSR 317 Java Persistence 2.0.

Implementing methods of abstract class - org.apache.olingo.odata2.jpa.processor.api. ODataJPATombstoneEntityListener

Step 1: Create a Java Class Create a Java class by extending "ODataJPATombstoneEntityListener.java".

Step 2: Implement the abstract method - getQuery Implement the abstract method getQuery. In the method implement the logic to generate a "javax. persistence.Query". The Query object can be created as shown below in the code snippet

```
if (!resultsView.getStartEntitySet().getName().equals(resultsView.getTargetEntitySet().getName())) {
  contextType = JPQLContextType.JOIN;
} else {
  contextType = JPQLContextType.SELECT;
}

JPQLContext jpqlContext = JPQLContext.createBuilder(contextType, resultsView).build();
JPQLStatement jpqlStatement = JPQLStatement.createBuilder(jpqlContext).build();

Query query = em.createQuery(jpqlStatement);
```

The JPQLStatement builder builds a JPQLStatement from the OData request (resultsView). The generated JPQLStatement can be enhanced to introduce conditions that filters and fetches delta JPA Entities. To enhance the JPQLStatement and add a condition to the where clause refer to the following code snippet.

```
String deltaToken = ODataJPATombstoneContext.getDeltaToken();
Query query = null;

if (deltaToken != null) {
  String statement = jpqlStatement.toString();
  String[] statementParts = statement.split(JPQLStatement.KEYWORD.WHERE);
  String deltaCondition = jpqlContext.getJPAEntityAlias() + ".creationDate >= {ts '" + deltaToken + "'}";

  if (statementParts.length > 1)
  {
    statement = statementParts[0] + JPQLStatement.DELIMITER.SPACE + JPQLStatement.KEYWORD.WHERE + JPQLStatement.
DELIMITER.SPACE +        deltaCondition + JPQLStatement.DELIMITER.SPACE + JPQLStatement.Operator.AND +
statementParts[1];
  } else {
    statement = statementParts[0] + JPQLStatement.DELIMITER.SPACE + JPQLStatement.KEYWORD.WHERE + JPQLStatement.
DELIMITER.SPACE + deltaCondition;
  }

  query = em.createQuery(statement);
}else
  query = em.createQuery(jpqlStatement.toString());
```

- Note:- It is up to the JPA application developers to come up with a logic suitable for use case.*

Step 3 - Implement the method generateDeltaToken Implement the method generateDeltaToken to generate a string representation of delta token. The delta token thus generated shall be used by client applications to fetch delta in their subsequent OData requests.

```
SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss.000");;
Date date = new Date(System.currentTimeMillis());
dateFormat.format(date);
return dateFormat.format(date);
```

Step 4 - Register the class as Entity Listener Register the above created class as an Entity Listener in JPA Entity.

```
@Entity
@Table(name = "T_SALESORDERHEADER")
@EntityListeners(com.sap.core.odata.processor.ref.jpa.listners.SalesOrderTombstoneListner.class)
public class SalesOrderHeader {
```

Implementing no methods of abstract class - org.apache.olingo.odata2.jpa.processor.api. ODataJPATombstoneEntityListener

At times JPA application developers would like to use pure JPA features like Callbacks to handle delta token. The library does provide support for use cases where the JPA application developer can write a call back method in the Entity Listener class and annotate them with javax.persistence.PostLoad. Such call back methods are executed by JPA providers in a stateless manner for every record fetched from the database table. In the call back method; logic can be written to decide whether a JPA entity fetched from Database table is a delta or not as shown below

Step 1 - Implement a JPA Entity Listener as per JPA 2.0 Specification Implement a JPA entity listener by extending ODataJPATombstoneEntityListener class and register the same with the JPA Entity. In the entity listener implement a method and annotate the same with Post Load annotation.

```
@PostLoad
public void handleDelta(Object entity) {
  SalesOrderHeader so = (SalesOrderHeader) entity;
  if (so.getCreationDate().getTime() < ODataJPATombstoneContext.getDeltaTokenUTCTimeStamp())
    return;
  else
    addToDelta(entity,ENTITY_NAME);
}
```

In the above code snippet addToDelta is a default implementation provided by ODataJPATombstoneEntityListener.