Seam Transaction

Seam Transaction Reference

1.1. Introduction

Unlike EJB session beans CDI beans are not transactional by default. Seam Transaction brings declarative transaction management to CDI beans by enabling them to use `@TransactionAttribute`. Seam also provides the `@Transactional` annotation, for environments where java EE APIs are not present.

1.2. Transaction Management

1.2.1. Configuration

In order to enable declarative transaction management for managed beans you need to list the transaction interceptor in beans.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="
    http://java.sun.com/xml/ns/javaee
    http://docs.jboss.org/cdi/beans_1_0.xsd">
  <interceptors>
    <class>org.jboss.seam.transaction.TransactionInterceptor</class>
  </interceptors>
</beans>
```

If you are in a Java EE 6 environment then you are good to go, no additional configuration is required.

Note

If you are deploying to JBoss AS6 it is important to know that it does not support meta data per bean archive and will throw a deployment error if defined twice. Additionally some Seam 3 modules such as Security already enable this Interceptor and defining it again will result in a deployment error.

This is not an issue with JBoss AS7 or Glassfish.

If you are not in an EE environment you may need to configure some things with Solder. You may need the following entries in your `beans.xml` file:

```
<beans xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:s="urn:java:ee"
  xmlns:t="urn:java:org.jboss.seam.transaction"
  xsi:schemaLocation="
    http://java.sun.com/xml/ns/javaee
    http://docs.jboss.org/cdi/beans_1_0.xsd">

    <t:SeSynchronizations>
      <s:modifies/>
    </t:SeSynchronizations>

    <t:EntityTransaction>
      <s:modifies />
    </t:EntityTransaction>

</beans>
```

Let's look at these individually.

```
<t:SeSynchronizations>
  <s:modifies/>
</t:SeSynchronizations>
```

Seam will attempt to use JTA synchronizations if possible. If not then you need to install the `SeSynchronzations` bean to allow seam to handle synchronizations manually. Synchronizations allow Seam to respond to transaction events such as `beforeCompletion()` and `afterCompletion()`, and are needed for the proper operation of the Seam Managed Persistence Context.

```
<t:EntityTransaction>
  <s:modifies />
</t:EntityTransaction>
```

By default Seam will attempt to look up `java:comp/UserTransaction` from JNDI (or alternatively retrieve it from the `EJBContext` if a container managed transaction is active). Installing `EntityTransaction` tells Seam to use the JPA `EntityTransaction` instead. To use this you must have a Seam Managed Persistence Context (see the Seam Persistence documentation for details) installed with qualifier`@Default`.

If your entity manager is installed with a different qualifier, then you need to use the following configuration (this assumes that `my` has been bound to the namespace that contains the appropriate qualifier, see the Solder documentation for more details):

```
<t:EntityTransaction>
  <s:modifies />
  <t:entityManager>
    <my:SomeQualifier/>
  </t:entityManager>
</t:EntityTransaction>
```

Note

You should avoid `EntityTransaction` if you have more than one persistence unit in your application. Seam does not support installing multiple `EntityTransaction` beans, and the `EntityTransaction` interface does not support two phase commit, so unless you are careful you may have data consistency issues. If you need multiple persistence units in your application then we highly recommend using an EE 6 compatible server, such as JBoss AS7.

1.2.2. Declarative Transaction Management

Seam adds declarative transaction support to managed beans. Seam re-uses the EJB `@TransactionAttribute` for this purpose, however it also provides an alternative `@Transactional` annotation for environments where the EJB API's are not available. An alternative to`@ApplicationException`, `@SeamApplicationException` is also provided. Unlike EJBs, managed beans are not transactional by default, you can change this by adding the `@TransactionAttribute` to the bean class.

Unlike in Seam 2, transactions will not roll back whenever a non-application exception propagates out of a bean, unless the bean has the transaction interceptor enabled.

If you are using seam managed transactions as part of the seam-faces module you do not need to worry about declarative transaction management. Seam will automatically start a transaction for you at the start of the faces request, and commit it before the render response phase.

Warning

`@SeamApplicationException` will not control transaction rollback when using EJB container managed transactions. If you are in an EE environment then you should always use the EJB API's, namely `@TransactionAttribute` and`@ApplicationException`.

Note

`TransactionAttributeType.REQUIRES_NEW` and `TransactionAttributeType.NOT_SUPPORTED` are not yet supported on managed beans.

Let's have a look at some code. Annotations applied at a method level override annotations applied at the class level.

```
@TransactionAttribute /*Defaults to TransactionAttributeType.REQUIRED */
        class TransactionalBean
        {

        /* This is a transactional method, when this method is called a transaction
        * will be started if one does not already exist.
        * This behavior is inherited from the @TransactionAttribute annotation on
        * the class.
        */
        void doWork()
        {
        ...
        }

        /* A transaction will not be started for this method, however it */
        /* will not complain if there is an existing transaction active. */
        @TransactionAttributeType(TransactionAttributeType.SUPPORTED)
        void doMoreWork()
        {
        ...
        }

        /* This method will throw an exception if there is no transaction active when */
        /* it is invoked. */

        @TransactionAttributeType(TransactionAttributeType.MANDATORY)
        void doEvenMoreWork()
        {
        ...
        }

        /* This method will throw an exception if there is a transaction active when */
        /* it is invoked. */
        @TransactionAttributeType(TransactionAttributeType.NOT_SUPPORTED)
```

```
void doOtherWork()
{
...
}
}
```

1.2.3. ServletRequestListener

Seam Transaction has a built in `ServletRequestListener` which automatically begins and commits (or rolls back if the transaction is set to rollback) a transaction for each request! This should end having to manually specify transactions, or wonder if a transaction is inplace.

Tip

Should the need arise for disabling this listener, a context param in web.xml named `org.jboss.seam.transaction.disableListener` set to `true` will disable the listener.