Flow Managed Persistence

Summary

Most applications access data in some way. Many modify data shared by multiple users and therefore require transactional data access properties. They often transform relational data sets into domain objects to support application processing. Web Flow offers "flow managed persistence" where a flow can create, commit, and close a object persistence context for you. Web Flow integrates both Hibernate and JPA object persistence technologies.

Apart from flow-managed persistence, there is the pattern of fully encapsulating PersistenceContext management within the service layer of your application. In that case, the web layer does not get involved with persistence, instead it works entirely with detached objects that are passed to and returned by your service layer. This chapter will focus on the flow-managed persistence, exploring how and when to use this feature.

Description

FlowScoped PersistenceContext

This pattern creates a PersistenceContext in flowScope on flow startup, uses that context for data access during the course of flow execution, and commits changes made to persistent entities at the end.

This pattern provides isolation of intermediate edits by only committing changes to the database at the end of flow execution. This pattern is often used in conjunction with an optimistic locking strategy to protect the integrity of data modified in parallel by multiple users. To support saving and restarting the progress of a flow over an extended period of time, a durable store for flow state must be used. If a save and restart capability is not required, standard HTTP session-based storage of flow state is sufficient. In that case, session expiration or termination before commit could potentially result in changes being lost.

To use the FlowScoped PersistenceContext pattern, first mark your flow as a persistence-context:

```
<?xml version="1.0" encoding="UTF-8"?>
<flow xmlns="http://www.springframework.org/schema/webflow"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.springframework.org/schema/webflow
http://www.springframework.org/schema/webflow/spring-webflow-2.0.xsd">
        <persistence-context />
</flow>
```

Then configure the correct FlowExecutionListener to apply this pattern to your flow. If using Hibernate, register the HibernateFlowExecutionListener. If using JPA, register the JpaFlowExecutionListener.

```
<webflow:flow-executor id="flowExecutor"
                flow-registry="flowRegistry">
        <webflow:flow-execution-listeners>
                <webflow:listener ref="jpaFlowExecutionListener" />
        </webflow:flow-execution-listeners>
</webflow:flow-executor>

<bean id="jpaFlowExecutionListener"
class="org.springframework.webflow.persistence.JpaFlowExecutionListener">
  <constructor-arg ref="entityManagerFactory" />
  <constructor-arg ref="transactionManager" />
</bean>
```

To make flow commit at the time when Flow finishes, enter the commit property of end-state.

```
<end-state id="bookingConfirmed" commit="true" />
```

That's it.

Now when Flow starts, the listener allocates new EntityManager to flowScope and control.
When accessing the data that occurs using the spring-based data access object in the Flow, this EntityManager is used automatically.
This data access operation is not the transaction processing target for keeping the independence of intermediate updating contents and should be executed only in read only transaction.

Reference

- Spring Web Flow reference 2.0.x
- Spring Web-Flow Framework Reference beta with Korean (by Park Chan Wook)
- Whiteship's Note