tapestry-jpa guide

Tapestry JPA provides integration between Tapestry 5 and the JPA Specifications (JPA1: JSR-220 and JPA2: JSR-317). It is modeled after tapestry's hibernate module and provides the same features. It has been used in production for some time now and is considered to be stable.

Setup

Requirements

In order to start using Tapestry JPA, you need to meet the following requirements.

- working persistence.xml with defined PersistenceUnit as you would have in any other JPA project
- a implementation of JPA2 in your classpath. We use EclipseLink 2.0.0 in our examples, but any other provider should work as well.
- a tapestry-ioc module for tapestry-jpa-core or a tapestry webapp for the extended features of tapestry-jpa

Dependencies

You need the following modules in order to use the JPA functionality.

- tapestry-jpa (for use with tapestry web applications)
- tapestry-jpa-core (for use with tapestry-ioc modules)

Maven2 example:

```
<dependency>
 <groupId>org.tynamo</groupId>
 <artifactId>tapestry-jpa</artifactId>
 <version>2.0.0</version>
</dependency>
<dependency>
 <groupId>org.tynamo</groupId>
 <artifactId>tapestry-jpa-core</artifactId>
 <version>2.0.0</version>
</dependency>
```

These artifacts are available in the maven main repository. No need to configure any additional repositories.

> **"JPAv2"**
> Please note that tapestry-jpa:2.0.0 and tapestry-jpa-core:2.0.0 do work with JPA2 only. There is a 1.0.0-SNAPSHOT module available in the snapshot repository http://ci.repository.codehaus.org, but it is not as stable as 2.0.0. Because of the limited possibilities of JPA1, it will never have as many features as the JPA2 module.

Configuration

For the module to work, you first need to tell tapestry-jpa what PersistenceUnit to use.

```
public static void contributeApplicationDefaults(
          MappedConfiguration<String, String> configuration)    {
     // ...
     configuration.add(JPASymbols.PERSISTENCE_UNIT, "nameOfPersistenceUnit");
}
```

That should be everything needed.

How to use Tapestry JPA

Injecting the EntityManager

In any Service, Page or Component you may inject an `EntityManager` like in the following example:

```
public class MyPage() {
    @Inject private EntityManager em;
    @Property private MyEntity entity;
    // code

    void onActivate(long id) {
       entity = em.find(MyEntity.class, id);
    }
}
```

Committing with tapestry-jpa

You may manage your transactions manually by retrieving a transaction using `em.getTransaction()`. But to make your life easier, you can use the `@CommitAfter` annotation.

```
public class MyPage() {
    @Inject private EntityManager em;
    // code

    @CommitAfter
    void onAction(long id) {
       MyEntity entity = em.find(MyEntity.class, id);
       em.remove(entity);
    }
}
```

Annotating a method with `@CommitAfter` will automatically commit the transaction after successfull execution. It is rollbacked if an exception is thrown.

Committing with tapestry-jpa-core

If you are using tapestry-jpa-core in a tapestry-ioc module, the @CommitAfter does not work. To make it work, you need to do something like this:

```
@Match("*DAO")
    public static void adviseTransactions(JPATransactionAdvisor advisor,
MethodAdviceReceiver receiver)   {
        advisor.addTransactionCommitAdvice(receiver);
    }
```

Note the `@Match` annotation which says what interfaces should be checked for the `@CommitAfter` annotation. Other useful values could be "*Service". In this case all interfaces named `UserService`, `ArticleService`, `CustomerService` would be matched.

Because tapestry-ioc is not able to check the annotations inside the implementation, it must be in the interface!