JPA EntityManager详解(一)

持久化上下文（Persistence Contexts）的相关知识，内容包括如何从Java EE容器中创建EntityManager对象、如何从Java SE中创建EntityManager对象、持久化上下文与事务（Transction）的关系，以及实体管理器工厂（Entity Manager Factory）的相关内容。

通过本章的学习，读者将深入掌握JPA中有关持久化上下文、事务处理的相关知识，从而能够更加深入地应用JPA。

11.1 获得EntityManager对象

那么如何获得EntityManager对象呢？这又是JPA中另外一个很重要的问题。

11.1.1 Java EE环境与J2SE环境

在详细讲述EntityManager对象之前，读者首先要分清楚两个概念，即Java EE环境与J2SE环境。因为在本章后面的学习中要经常提到这两个概念，所以读者一定要先理解它们，为以后的学习打好基础。

— Java EE环境，包括EJB容器和Web容器。

（1）Web容器：只运行Web应用的容器，例如Tomcat就是开源的Web容器，它可以运行JSP、Servlet等。

（2）EJB容器：运行在EJB组件的容器，提供EJB组件的状态管理、事务管理、线程管理、远程数据资源访问、连接管理和安全性管理等系统级服务。例如JBoss为EJB容器和Web容器（Web容器是集成了Tomcat）结合。

部署在EJB容器中的JAR包都可以认为是运行在EJB容器中。但JBoss中的Web应用，比如war包中的类就不是运行在EJB容器中，而是运行在Web容器中。

— J2SE环境

最普通Java运行环境，例如一个HelloWorld的Java程序就是运行在J2SE的环境中，通常使用main入口方法作为程序启动的触发。

如图11-1所示，它说明了Java EE与J2SE环境的关系。

11.1.2 两种类型的EntityManager对象

根据EntityManager对象的管理方式，可以有以下两种类型。

— 容器托管的（container-managed）EntityManager对象

容器托管的EntityManager对象最简单，程序员不需要考虑EntityManager连接的释放，以及事务等复杂的问题，所有这些都交 给容器去管理。容器托管的EntityManager对象必须在EJB容器中运行，而不能在Web容器和J2SE的环境中运行。本书前面讲述的 EntityManager对象都是通过注入 @PersistenceContext注释来获得的，其实，这种获得EntityManager对象的方式就是容器托管的。

— 应用托管的（application-managed）EntityManager对象

应用托管的EntityManager对象，程序员需要手动地控制它的释放和连接、手动地控制事务等。但这种获得应用托管的 EntityManager对象的方式，不仅可以在EJB容器中应用，也可以使 JPA脱离EJB容器，而与任何的Java环境集成，比如说Web容器、J2SE环境等。所以从某种角度上来说，这种方式是JPA能够独立于EJB环境运 行的基础。

理想状态下，最好是选用容器托管的EntityManager对象的方式，但在特殊的环境下，还是需要使用应用托管的EntityManager对象这种方式。

正是因为应用托管的EntityManager对象的连接释放、事务控制比较复杂，所以在使用时涉及的相关内容比较多，这些内容将在本章后面部分详细讲述，这里读者应对两种方式有一个大致的了解，两种EntityManager对象类型的比较如表11-1所示。

表11-1 容器托管与应用托管的EntityManager对象对比

比较内容

容器托管的（container-managed）EntityManager对象

应用托管的（application-managed）EntityManager对象

获得方式

两种方式：1 @PersistenceContex注入 2 JNDI获得

EntityManagerFactory创建

支持事务

JTA

JTA、RESOURCE\_LOCAL

运行环境

EJB容器

EJB容器、Web容器、J2SE环境

11.1.3 容器托管的（container-managed）EntityManager对象

容器托管的EntityManager对象只能运行在EJB容器中。所以可以这样理解，只有在EJB-JAR包中，才可以获得容器托管的EntityManager对象，否则只能获得应用托管的EntityManager对象。

在EJB容器中获得EntityManager对象主要有两种方式，即@PersistenceContext注释注入和JNDI方式获得。

11.1.3.1 通过@PersistenceContext注释注入

这种方式获得EntityManager对象最为常用，例如下面代码所示。

@Stateless

public class CustomerService implements ICustomerService {

@PersistenceContext(unitName = "jpaUnit")

private EntityManager entityManager;

public List<CustomerEO> findAllCustomers() {

Query query = entityManager.createQuery("SELECT c FROM CustomerEO c");

List<CustomerEO> result = query.getResultList();

for (CustomerEO c : result) {

System.out.println(c.getId()+","+c.getName());

}

return result;

}

}

在使用此种方式创建EntityManager对象时，需要注意以下几个问题。

— @PersistenceContext注释中，其中unitName为persistence.xml文件中<persistence-unit>元素中的属性“name”的值，表示要初始化哪个持久化单元，如下所示。

Xml代码

<persistence>

<persistence-unit name="jpaUnit" transaction-type="JTA">

</persistence-unit>

</persistence>

— @PersistenceContext注释中还可以配置其他的设置，它的定义如下所示。

Java代码

@Target({TYPE, METHOD, FIELD}) @Retention(RUNTIME)

public @interface PersistenceContext{

String name() default "";

String unitName() default "";

PersistenceContextType type default TRANSACTION;

PersistenceProperty[] properties() default {};

}

— 其中PersistenceContextType可以设置创建EntityManager对象时，持久化上下文的作用范围，它的意义在于对有状态的Bean（Stateless Bean）可以跨事务操作实体。它主要有两种方式，定义如下所示。

Java代码

public enum PersistenceContextType {

TRANSACTION,

EXTENDED

}

默认情况下使用TRANSACTION，有关TRANSACTION方式和EXTENDED方式创建EntityManager对象的异同，将在下文中详细讲述，这里读者简单了解一下即可。

11.1.3.2 通过JNDI的方式获得

如果指定了@PersistenceContext注释中的name值，则设置了持久化上下文的JNDI名称。通过SessionContext可以创建EntityManager对象。

例如，下面代码为通过JNDI方式获得EntityManager对象。

@Stateless

@PersistenceContext(name="jpa")

public class CustomerService implements ICustomerService {

@Resource

SessionContext ctx;

public List<CustomerEO> findAllCustomers() {

EntityManager entityManager = (EntityManager) ctx.lookup("jpa");

Query query = entityManager.createQuery("SELECT c FROM CustomerEO c");

List<CustomerEO> result = query.getResultList();

for (CustomerEO c : result) {

System.out.println(c.getId()+","+c.getName());

}

return result;

}

}

11.1.4 应用托管的（application-managed）EntityManager对象

应用托管的EntityManager对象，不仅可以在Java EE环境中获得，也可以应用在J2SE的环境中。但无论是在什么情况下获得的EntityManager对象，都是通过实体管理器工厂 （EntityManagerFactory）对象创建的。所以如何获得应用托管的EntityManager对象关键是 EntityManagerFactory对象如何获得。

下面就分别讲述在EJB容器、Web容器和J2SE环境中如何获得EntityManagerFactory对象。

11.1.4.1 EJB容器中获得

在EJB容器中，EntityManagerFactory对象可以通过使用注入@PersistenceUnit注释获得，例如下面代码为在EJB容器中，获得应用托管的EntityManager对象的方法。

@Stateless

public class CustomerService implements ICustomerService {

@PersistenceUnit(unitName="jpaUnit")

private EntityManagerFactory emf;

public List<CustomerEO> findAllCustomers() {

/\*\*创建EntityManager对象\*/

EntityManager em = emf.createEntityManager();

Query query = em.createQuery("SELECT c FROM CustomerEO c");

List<CustomerEO> result = query.getResultList();

for (CustomerEO c : result) {

System.out.println(c.getId()+","+c.getName());

}

/\*\*关闭EntityManager \*/

em.close();

return result;

}

}

通过以上的EntityManager对象代码，可以总结出以下几个问题。

— 应用托管的EntityManager对象，要在代码中手动地创建和关闭，例如下面代码所示。

EntityManager em = emf.createEntityManager();

/\*\*其他的业务逻辑\*/

em.close();

这点正是与容器托管的EntityManager对象的最大不同之处。事实上，容器托管的EntityManager对象，它的创建和关闭是由容器负责管理的，所以不需要编写代码来控制。

— 应用托管的EntityManager对象，都是通EntityManagerFactory对象来创建的。在容器中可以通过使用注入@PersistenceUnit注释的方法实现，它的定义如下所示。

@Target({TYPE, METHOD, FIELD}) @Retention(RUNTIME)

public @interface PersistenceUnit{

String name() default "";

String unitName() default "";

}

其中，属性unitName为persistence.xml文件中<persistence-unit>元素中的属性“name”的值，表示要初始化哪个持久化单元，与@PersistenceContext注释中unitName属性相同。

11.1.4.2 Web容器中获得

在Web容器中，EntityManagerFactory对象也可以通过使用注入@PersistenceUnit注释获得。例如，下面代码为在 Servlet中，获得应用托管的EntityManager对象的方法。 /syntaxhighlighter/clipboard\_new.swf">

public class TestServlet extends HttpServlet {

@PersistenceUnit(unitName = "jpaUnit")

private EntityManagerFactory emf;

public TestServlet() {

super();

}

public void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

doPost(request, response);

}

public void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html");

PrintWriter out = response.getWriter();

out.println("<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.01 Transitional //EN\">");

out.println("<HTML>");

out.println(" <HEAD><TITLE>A Servlet</TITLE></HEAD>");

out.println(" <BODY>");

if (emf != null) {

/\*\*创建EntityManager 对象\*/

EntityManager entityManager = emf.createEntityManager();

try {

Query query = entityManager

.createQuery("SELECT c FROM CustomerEO c");

List<CustomerEO> result = query.getResultList();

for (CustomerEO c : result) {

System.out.println(c.getId() + "," + c.getName());

}

} finally {

/\*\*关闭EntityManager\*/

entityManager.close();

}

}

out.println(" </BODY>");

out.println("</HTML>");

out.flush();

out.close();

}

}